

# NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



19970311 014

## THESIS

**RE-ENGINEERING OF THE  
COMPUTER-AIDED PROTOTYPING SYSTEM  
FOR PORTABILITY**

by

Dennis M. Irwin

September, 1996

Thesis Advisors:

Luqi  
Berzins

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 1

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 1996.		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE RE-ENGINEERING OF THE COMPUTER-AIDED PROTOTYPING SYSTEM FOR PORTABILITY			5. FUNDING NUMBERS	
6. AUTHOR(S) Dennis M. Irwin				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words)  The Computer-Aided Prototyping System (CAPS) Release 1 currently runs only on SPARC workstations running SunOS version 4.1.3. This limits the usefulness of CAPS, particularly since Sun has announced it has no plans to continue support for SunOS version 4.x. A solution to this problem is to increase the portability of CAPS by first porting CAPS to the Solaris 2.5 operating system. Towards this end, this thesis discusses and evaluates the underlying system software and software tools necessary to build and run CAPS within the Solaris 2.5 operating environment for SPARC workstations. As a result of this effort, a version of CAPS has been created that runs on a SPARC workstations using the Solaris 2.5 operating system. Furthermore, the research has identified the necessary software tools and potential problem areas for determining the feasibility of porting CAPS to other platforms. Versions of X Windows, Motif, Synthesizer Generator, Eli, TAE Plus, and the VADSelf Ada compiler are required. Since TAE Plus only supports the SunAda (VADS) compiler, use of a different Ada compiler will require either porting the TAE Ada bindings or using an alternative to TAE Plus. Additionally, an explicit installation of Motif is required to provide all the libraries needed to produce static builds of the CAPS components.				
14. SUBJECT TERMS Software Engineering, Rapid Prototyping, CAPS, Solaris, XWindows, Motif, TAE, Ada, User Interface.			15. NUMBER OF PAGES 116	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)  
Prescribed by ANSI Std. Z39-18 298-102



Approved for public release; distribution is unlimited.

**RE-ENGINEERING OF THE COMPUTER-AIDED PROTOTYPING  
SYSTEM FOR PORTABILITY**

by

Dennis M. Irwin  
Lieutenant, United States Navy  
B.S., Louisiana State University, 1988

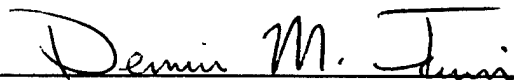
Submitted in partial fulfillment  
of the requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

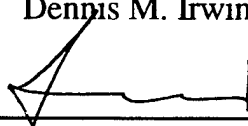
**NAVAL POSTGRADUATE SCHOOL  
September 1996**

Author:

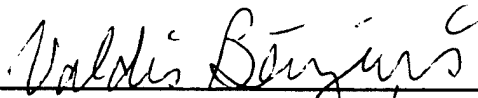


Dennis M. Irwin

Approved by:



Luqi, Thesis Advisor



Valdis Berzins, Thesis Advisor



Ted Lewis, Chairman  
Department of Computer Science



## ABSTRACT

The Computer-Aided Prototyping System (CAPS) Release 1 currently runs only on SPARC workstations running SunOS version 4.1.3. This limits the usefulness of CAPS, particularly since Sun has announced it has no plans to continue support for SunOS version 4.x. A solution to this problem is to increase the portability of CAPS by first porting CAPS to the Solaris 2.5 operating system.

Towards this end, this thesis discusses and evaluates the underlying system software and software tools necessary to build and run CAPS within the Solaris 2.5 operating environment for SPARC workstations.

As a result of this effort, a version of CAPS has been created that runs on a SPARC workstations using the Solaris 2.5 operating system. Furthermore, the research has identified the necessary software tools and potential problem areas for determining the feasibility of porting CAPS to other platforms. Versions of X Windows, Motif, Synthesizer Generator, Eli, TAE Plus, and the VADSself Ada compiler are required. Since TAE Plus only supports the SunAda (VADS) compiler, use of a different Ada compiler will require either porting the TAE Ada bindings or using an alternative to TAE Plus. Additionally, an explicit installation of Motif is required to provide all the libraries needed to produce static builds of the CAPS components.



## TABLE OF CONTENTS

I. INTRODUCTION .....	1
A. THE SOFTWARE ENGINEERING PROBLEM .....	1
B. RAPID PROTOTYPING .....	2
C. COMPUTER-AIDED PROTOTYPING SYSTEM (CAPS) .....	4
D. THESIS OBJECTIVE .....	4
E. OVERVIEW OF REMAINING CHAPTERS .....	5
II. COMPUTER-AIDED PROTOTYPING SYSTEM (CAPS) .....	7
A. CAPS BENEFITS .....	8
B. MAJOR CAPS COMPONENTS .....	9
1. Graph Editor .....	9
2. PSDL Editor .....	9
3. Expander .....	9
4. Translator .....	10
5. Scheduler .....	10
6. Tool Interface .....	10
C. OPERATING ENVIRONMENT COMPONENTS .....	10
1. Solaris 2.5 and OpenWindows 3.5 .....	11
2. X Windows and Motif .....	13
3. TAE Plus v5.31 .....	14



4.	Synthesizer Generator 4.2 .....	16
5.	Eli 4.4 .....	16
6.	VADSSelf™ 6.2 .....	17
7.	Sun SPARCworks 4.0 .....	17
8.	GNAT 3.05 .....	17
III. PORTING CAPS TO SOLARIS .....		19
A.	GENERAL STRATEGY .....	19
B.	SYSTEM REQUIREMENTS .....	19
C.	EVALUATION .....	20
1.	Solaris Compatibility Issues .....	20
2.	X-Windows and Motif .....	22
3.	TAE Plus .....	23
4.	Synthesizer Generator and Eli .....	23
5.	Compiling, Linking, and Loading .....	23
6.	Shell Scripts .....	24
IV. CONCLUSIONS AND FUTURE RESEARCH .....		27
A.	SUMMARY .....	27
B.	SUGGESTIONS FOR FUTURE WORK .....	29
1.	Alternatives to TAE Plus .....	29
2.	Ada 95 Compiler Integration .....	29

3.	Tools Interface .....	30
4.	Personal Computer Version .....	30
5.	Shell Scripts .....	30
6.	Alert Dialog Boxes .....	31
7.	New Environments .....	31
C.	CONCLUSION .....	32
LIST OF REFERENCES .....		33
BIBLIOGRAPHY .....		35
APPENDIX A. LOCAL SOLARIS SYSTEM INFORMATION .....		39
APPENDIX B. SHELL SCRIPTS .....		61
APPENDIX C. WORLD WIDE WEB SITES .....		91
APPENDIX D. POINTS OF CONTACT .....		99
INITIAL DISTRIBUTION LIST .....		103



## ACKNOWLEDGEMENT

Thanks go out to everyone in the Computer Science Department who assisted me in my thesis research. I would like to thank Sue Whalen for allowing me to camp out in her office and assisting me with system administration problems. I would also like to thank Mike Williams for attending to all my hardware needs. Finally, special thanks goes to my wife Annie for her undying support for me during my research effort and the troublesome write-up.



## **I. INTRODUCTION**

Today, computers are increasingly inexpensive and fast. They are becoming universally prevalent and are taking on the appearance of a household appliance. With the increased use of computers in everyday life, software designers face bigger problems. Software must be inexpensive to develop, meet user requirements, and be reliable. Consequently, there is a great need to improve software productivity and reliability [Ref. 1].

### **A. THE SOFTWARE ENGINEERING PROBLEM**

The problem facing software engineers today is how to produce and manage large-scale software projects involving millions of lines of code and teams of programmers. Such large-scale projects are not uncommon in the Department of Defense.

One of the problems in the development of large-scale software systems is requirements analysis and specification. As pointed out by Luqi [Ref. 1: p.111], "as systems get larger and serve more diversified user communities, formulating requirements that accurately represent the customer's needs becomes the limiting factor in producing useful software."

Typically, the customer gives the software engineer a set of requirements of what they believe the software should be. However, such a statement of requirement is often vague, incomplete, and inconsistent. The software engineer must analyze what the customer desires and build the software. It is at the end of a costly production and testing that the customer

receives the product. If the customer's needs have changed during development or the product is not what the customer had in mind, then the product has to be reworked at great expense.

The majority of errors in large software projects are due to faults in the requirements specification, which are not normally detected until after the product has been developed and in the testing phase. [Ref. 2] However, the cost of fixing software errors rises dramatically later in the life-cycle. Therefore, more effort needs to be placed earlier in the software development life-cycle to reduce errors and the cost of maintenance.

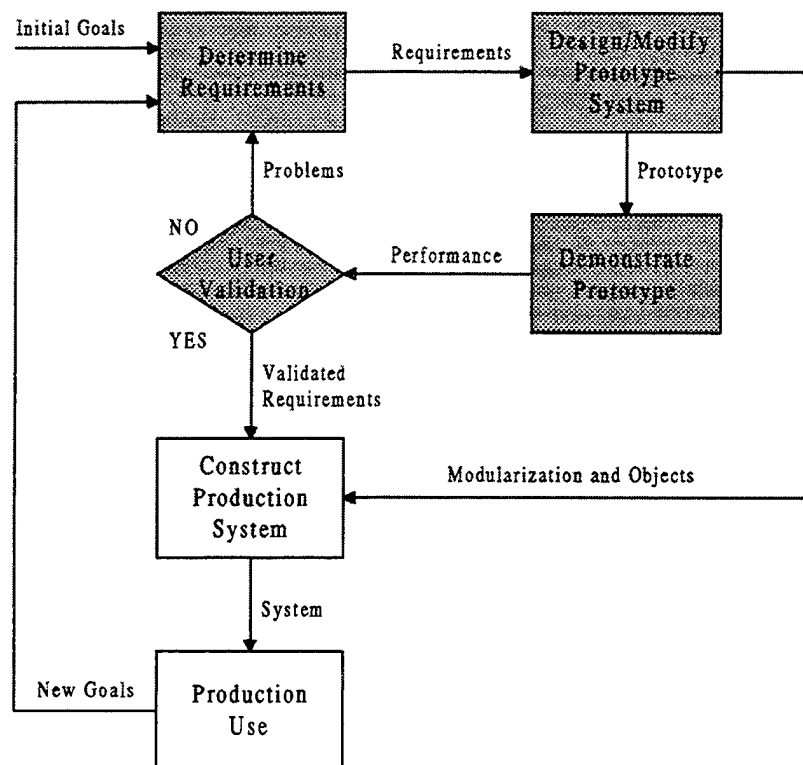
The need for more effective software engineering is clearly evident in the Department of Defense, where software technology lags far behind hardware development. In FY 1995, DoD's investment in software was \$42 billion, yet most large software projects are overbudget, overtime, or even failures. Specifically, 53% of the software projects had cost or time overruns, 31% were canceled, and only 16% were delivered on-time and on-budget. [Ref. 3]

## **B. RAPID PROTOTYPING**

Changes to a software system, whether due to changes in requirements, the need for improvements, or bug repairs, accounts for more than half of the total cost of the software. One solution to reducing this cost is through the use of prototyping. [Ref. 4: p. 13]

"...a prototype is a concrete executable model of selected aspects of a proposed system." [Ref. 4: p. 13] Prototyping involves an iterative approach to design with many opportunities for the customer to provide feedback before the design goes into production.

The software engineer uses the customers specifications to draw up the software's requirements, determines what the problem areas are, and produces a prototype to demonstrate the area in question. The customer has an opportunity to interact with the prototype and determine if it meets his or her needs. Inputs from the customer produces changes in the requirements and corresponding changes in the prototype. Only after the customer is satisfied with the prototype does the software goes into production. In a similar manner, the existing prototype can be modified to test proposed changes to the production software during the maintenance phase. Therefore, "rapid prototyping is the process of quickly building and evaluating a series of prototypes." [Ref. 4: p. 13] Figure 1, taken from [Ref. 4:



**Figure 1. The Prototyping Process.**



p. 14], illustrates the iterative prototyping process. However, prototyping is not practical unless it can be done rapidly, accurately, and cheaply [Ref. 1: p. 111].

### **C. COMPUTER-AIDED PROTOTYPING SYSTEM (CAPS)**

To assist software engineers in constructing and evaluating prototypes rapidly and systematically, software tools are introduced [Ref. 4: p. 14]. One of the tools available to software designers is the Computer-Aided Prototyping System (CAPS), an ongoing research project at the Naval Postgraduate School (NPS). CAPS provides the tools needed to perform rapid prototyping of real-time systems. CAPS supports rapid prototyping by providing an integrated set of automated programs generation and design decision support capabilities. CAPS accomplishes this by utilizing a graphical editor with decomposition, software reuse, and automatic code generation where possible. [Ref. 4]

### **D. THESIS OBJECTIVE**

In order to keep in step with technological advances in computing, this thesis examines the system requirements needed to upgrade CAPS. The next release of CAPS should not only incorporate improvements to the individual tools, but should also have increased portability and be adaptable to additional platforms. Towards this end, the thesis research ascertained and implemented the necessary system and software tools required to port CAPS to the Solaris 2.5 operating system for Sun SPARCstation. Furthermore, the new version incorporates improved CAPS components from Release 2.

As a result of this thesis research, CAPS Release 2 will be available for SPARC workstations running either SunOS 4.1.3 or Solaris 2.5 operating systems. Additionally, this porting effort identified the software requirements and potential problem areas that will be

useful in porting CAPS to additional operating environments. This effort is an enabling step for a major technology transfer effort to be undertaken by the research group to make computer-aided prototyping technology available through out the Department of Defense (DoD).

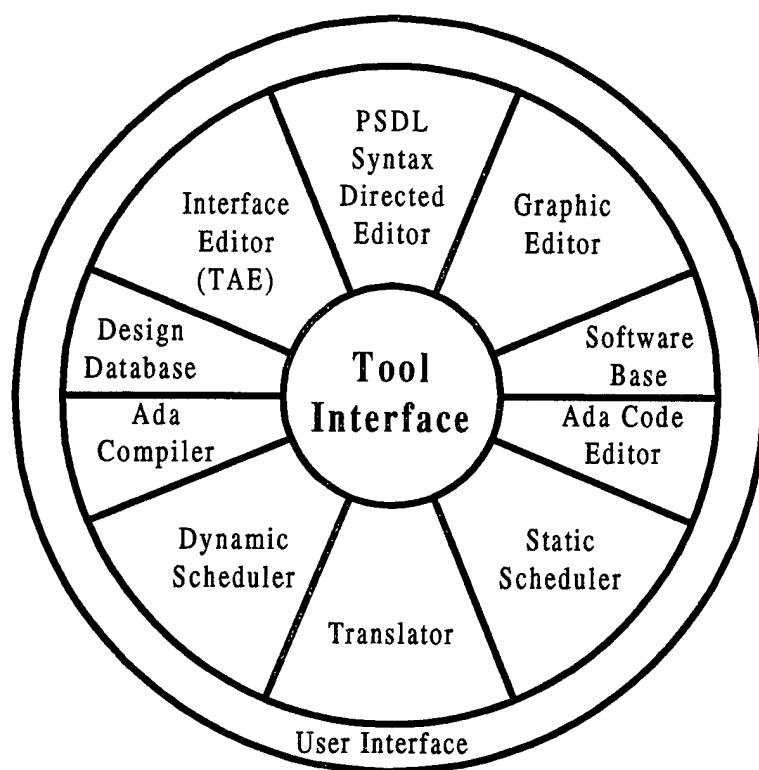
## **E. OVERVIEW OF REMAINING CHAPTERS**

This chapter discussed the software engineering problem, rapid prototyping as a possible solution, and CAPS as a tool for rapid prototyping of real-time systems. The remainder of the thesis consists of the following chapters. Chapter II describes CAPS in more detail, including the system architecture. Chapter III describes the effort to port CAPS to Solaris, including the software required to build and run CAPS on a Solaris system. Chapter IV summarizes results and provides future work to continue the porting of CAPS to other operating environments. Appendix A provides specific information on the local Solaris installation and system maintenance. Appendix B contains the CAPS shell scripts. Appendix C provides a list of World Wide Web Pages useful for additional subject information. Finally, Appendix D provides a list of points of contacts for hardware and software products.



## II. COMPUTER-AIDED PROTOTYPING SYSTEM (CAPS)

CAPS is an environment for rapidly developing prototypes for real-time systems based on the Prototype System Description Language (PSDL) [Ref. 2]. CAPS is actually an integrated set of development tools with a common user interface as illustrated in Figure 2, taken from [Ref. 5: p. 204].



**Figure 2. The CAPS Tool Interface.**

## **A. CAPS BENEFITS**

CAPS is public domain software capable of automatically producing fast and reliable code, thus providing a low cost solution for software engineers. In general, CAPS [Ref. 3]:

- automates software development,
- improves software quality,
- reduces development time,
- decreases life-cycle costs,
- supports Mil-Std 498 for software acquisition and development,
- aids in the DoD Software Reuse Initiative (SRI), and
- provides better overall project management.

Some of the benefits CAPS provides the user are [Ref. 6]:

- graphic model representation of the software design,
- non-procedural annotations, requiring no specific programming language knowledge during the design and specification phase,
- automatic generation of code free of syntax errors and interface consistency errors,
- automatic generation of schedules to meet strict real-time deadlines,
- support for computer-aided generation of graphical user interfaces and simple animations of prototype behavior,
- fast and early feedback to the user through an executable prototype,
- early detection of errors in the requirements phase,

- the ability to firm up requirements before production through iterative assessment and modification of graphical representations, as well as requirements changes after delivery,
- easy modification of software designs to meet the customer's changing needs,
- requirements tracing through facilities for recording dependencies, linking requirements to specification, and preserving design information, and
- computer-aided assessment of hardware/software tradeoffs relative to different types of hardware.

## **B. MAJOR CAPS COMPONENTS**

The current release of CAPS is composed of six main components which communicate with each other directly or through the operating system via a collection of shell scripts.

### **1. Graph Editor**

The graph editor is used to create CAPS enhanced data flow diagrams, which will be converted into PSDL descriptions by the PSDL Editor.

### **2. PSDL Editor**

The PSDL editor incorporates the PSDL Syntax Directed Editor (SDE). The SDE takes information from the data flow diagrams created with the graph editor and allows the user to edit the PSDL descriptions. The SDE is created using the Synthesizer Generator.

### **3. Expander**

Expands the PSDL source.

#### **4. Translator**

The translator converts the PSDL code representation of the prototype into Ada packages. Primarily, it implements the data streams and control constraints described in the PSDL source. The translator is built using Eli.

#### **5. Scheduler**

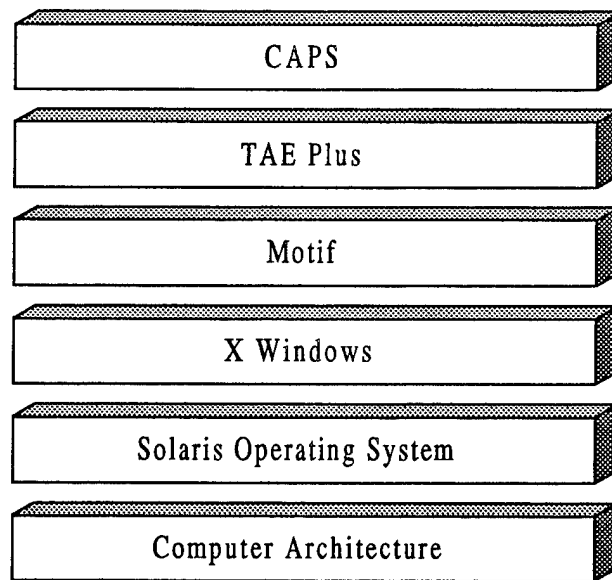
The scheduler incorporates scheduling algorithms to automatically construct real-time schedules, if feasible, for the prototypes. The scheduler performs both static and dynamic scheduling. The static scheduler precomputes and allocates time slots for time-critical operations based on a worst case execution time. The dynamic scheduler controls non-time-critical operations during runtime by allowing them to run in time slots not being used by time-critical operations. [Ref. 4]

#### **6. Tool Interface**

The tool interface provides a graphical user interface for accessing all of the CAPS tools. The current release only supports the designer mode interface (caps\_d.exe). The user interface was created using TAE Plus.

### **C. OPERATING ENVIRONMENT COMPONENTS**

The Solaris 2.5 version of CAPS is built upon a layered operating environment as shown in Figure 3. With Solaris available for many different platforms, it is envisioned that upper layers of this environment can be ported relatively easy to different platforms.



**Figure 3. System Architecture Layers.**

### **1. Solaris 2.5 and OpenWindows 3.5**

Solaris 2.5 (sometimes referred to as SunOS 5.5) is the latest version of the Unix operating system available at the time of this writing from Sun Microsystems and is being shipped with their new workstations. Solaris is based on AT&T Unix System V Release 4 (SVR4).

There is a risk of confusion with regard to the naming conventions of Sun's operating systems. Solaris 2.x is the same as SunOS 5.x. Sometimes the older SunOS 4.x is referred to as Solaris 1.x. In this document, SunOS refers to SunOS 4.1.x and Solaris refers to SunOS 5.x.



Solaris is a true 32-bit, multi-threaded, multi-processor operating system with implementations available for multiple platforms including SPARC, x86/Pentium/Pentium Pro, and Power PC. It also includes enhancements to take advantage of the higher performance available from Sun's new UltraSPARC processor (64-bit). [Ref. 7]

Solaris is used worldwide with an installed base of over two million users and supports for over 10,000 applications for SPARC systems and over 1,000 applications for Solaris x86. [Ref. 7]

Solaris provides many advantages including portability, scalability, interoperability, and compatibility. Solaris also include added functionality not found in SVR4, such as symmetric multi-processing with multi-threads, real-time functionality, increased security, and improved system administration. [Ref. 8]

Real-time functionality is supported in Solaris 2.5 through the use of a real-time scheduling policy in addition to a timesharing policy. Users can set real-time priorities for a process so that it will get the CPU as soon as it is ready to run. [Ref. 9]

One of the key features of Solaris is its availability for several different architectures, including personal computers based on Intel processors (Solaris x86).

Solaris 2.5 supports multiple processor architectures from a single source tree, preserving close compatibility between binary applications. This means developers do not have to rewrite application source code for each hardware architecture. Instead of developing three or more versions of the same applications, ISVs (Independent Software Vendors) can create one source version, compile the binaries on various hardware systems, and deploy the application across multiple architectures. Solaris 2.5 also provides access to new computing technologies such as the Java programming language... [Ref. 10]

Therefore, Solaris provides the potential of porting CAPS to a variety of platforms running the Solaris operating system.

## **2. X Windows and Motif**

The X Consortium's X Windows System and the Open Software Foundation (OSF) Motif provide the graphical user interface for many Unix-based applications. Currently, the Solaris operating system ships with the X Windows (X11R5) and Motif 1.2.4 runtime libraries. The latest stand alone releases are X11R6/R6.1 and Motif 2.0.

X Windows, which was originally developed at MIT, "... is a network-transparent window system which runs on a wide range of computing and graphics machines." [Ref. 11] Additionally, the X Window standard has enabled applications with graphics to become independent of the system hardware and software, thereby increasing portability.

The X Window System allows developers to write applications that can display information (graphical or otherwise) and accept input on one device while running on a different computer in the network. In X terminology, the application logic, written by the developer, is called the client. The client is linked with the MIT X windows libraries and the libraries of a specific X-based toolkit (such as Motif) to create a complete application. A client can reside on a remote computer anywhere in a network. The display logic of X, called the server, resides on an individual user's computer and is provided by the machine's vendor or a third-party software vendor. A client and server can also reside on the same machine. To ensure software portability and compatibility, all X servers must conform to the X Window System X11 protocols. [Ref. 12]

OSF/Motif provides a standardized graphical user interface for applications running on the X Window System. Motif provides a PC-style behavior and appearance that can be run on many different platforms supporting the X Window System (specifically X11R5). One

of the main components provided by Motif is its User Interface Toolkit, which is based on the X11 Intrinsics. The toolkit provides the developer with a set of graphical objects (widgets), such as menus and scroll bars, which can be used to build graphical user interfaces with a consistent style. [Ref. 13]

With the increased popularity of network computing environments, the X Window System and Motif are being used on more platforms than just workstations and X terminals.

### **3. TAE Plus v5.3l**

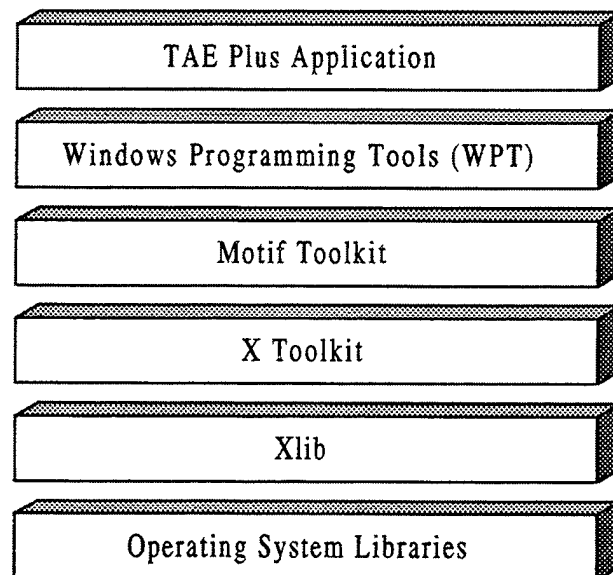
“TAE (Transportable Applications Environment) Plus is a portable software development environment that supports rapid building, tailoring, and management of graphic-oriented user interfaces.” [Ref. 14: p. 1] TAE Plus version 5.3 is the first commercial release of TAE Plus from Century Computing, Inc. [Ref. 15: p. 1] TAE Plus was originally developed for NASA’s Goddard Space Flight Center.

TAE Plus provides a WorkBench for the design and layout of an application’s user interface, Window Programming Tools (Wpt) Package which is a set of callable subroutines used to control the user interface during runtime, and Code Generator which automatically generates code for the user interface in C, C++, or Ada. [Ref. 14: p. 1] TAE Plus uses MIT’s X Window System and Open Software Foundation’s Motif Toolkit. [Ref. 14: p. 4]

TAE Plus is based on the X Window System and the Motif Toolkit. [Ref. 14: p. 9] However, TAE Plus hides the complexities of the underlying X Windows and Motif, making it easy for even non-programmers to develop graphical user interfaces in a Unix environment. TAE Plus accomplishes this through Wpt package calls. The Wpt package is layered on top

of the Motif Toolkit which provides high level interface objects called “widgets”. The Motif Toolkit interfaces with the X Toolkit, which is based on Xlib that provides the graphic primitives for X Windows. The X Windows server interfaces between the X Windows System and the underlying operating system and hardware. [Ref. 14: p. 10] Figure 4, taken after [Ref. 14], illustrates this relationship.

Additionally, TAE Plus creates a user interface definition that is separate from the application code, thereby allowing changes to the interface without having to changing the application code. [Ref. 14: p. 17]



**Figure 4. TAE Plus, Motif, and X11.**

#### **4. Synthesizer Generator 4.2**

One of the tools used to build the CAPS PSDL Editor is the Synthesizer Generator from GrammaTech, Inc.

The Synthesizer Generator is a tool for implementing language-based editors. The editor designer prepares a specification that includes rules defining a language's context-free abstract syntax, context-sensitive relationships, display format, and concrete input syntax. From this specification, the Synthesizer Generator creates an editor for manipulating objects according to these rules. [Ref. 16: p. 1]

"The Synthesizer Generator is especially well suited for creating editors that enforce the syntax and semantics of a particular language." [Ref. 16: p. 1] In this case, the language is the CAPS Prototyping System Description Language (PSDL). The Synthesizer Generator is used to generate the Syntax-Directed Editor (SDE) within the PSDL Editor.

#### **5. Eli 4.4**

Another tool used to build CAPS is Eli. Eli is being developed by the Compiler Tools Group Department of Electrical and Computer Engineering at the University of Colorado and is freely available. Eli is a set of tools that provide a sophisticated compiler construction environment which can be used to automatically generate complete language implementations from specifications. The compiler construction methods and techniques utilized by Eli can be applied to a variety of problems other than just the development of compilers for programming languages. Eli produces code that is comparable to good code done by hand, but at generally at one-third the time. In this case, Eli is used in the build of the CAPS Translator module. [Ref. 17]

## **6. VADSSelf™ 6.2**

VADSSelf™ is an Ada83 compiler available from Rational Software Corporation. The original version of CAPS was created with and utilized the Sun Ada compiler version 1.1. The Sun Ada compiler is actually the Verdex Ada compiler marketed by Sun. The Verdex compiler was acquired by Rational; therefore, VADSSelf™ 6.2 is the latest version of the Sun Ada/Verdex compiler family and is compatible with CAPS.

## **7. Sun SPARCworks 4.0**

SPARCworks version 4.0 for Solaris was the primary C/C++ compiler used to build the tools needed to build CAPS and to build CAPS itself.

## **8. GNAT 3.05**

The GNAT is a Ada95 compiler that is part of the GCC compiler system. GNAT is available for free from New York University and the Free Software Foundation (<ftp://cs.nyu.edu/pub/gnat/>). GNAT itself was not used to build CAPS, but the GCC C/C++ compiler (version 2.7.2 ) which is part of it was used to build parts of the previously mention tools when the SPARCworks compiler proved to be too strict for the older C source code.



### III. PORTING CAPS TO SOLARIS

#### A. GENERAL STRATEGY

The research consisted of the following stages. The first was to determine the major pieces of software that were required to setup the operating environment needed to build and run CAPS. The second step was to obtain and install Solaris versions of the required software, either from source code or pre-compiled binaries. Next, incompatibilities were determined by attempting to run CAPS Release 1 on the new Solaris environment. Finally, the new CAPS components (Release 2) were statically compiled for the Solaris 2.5 environment and the shell scripts were modified as necessary.

An extensive portion of the research involved acquiring, installing, and configuring the Solaris 2.5 operating system on two older Sun workstations: *perseus*, a SPARCstation 2, and *aldebaran*, a SPARCstation 1+. Later, the operating system was installed on *suns8*, a SPARCstation 10, and *suns9*, a SPARCstation 20, with *suns8* configured to export software to the other Solaris workstations. The systems were connected to the Computer Science Department network, but were configured stand alone due to the fact that they were research machines and access was restricted.

#### B. SYSTEM REQUIREMENTS

CAPS Release 1 runs on a Sun SPARCstation running SunOS 4.1.1 or later with X Windows (X11R4 or X11R5), OSF/Motif 1.1.2 or later, SunAda Compiler 1.1, and optionally TAE Plus v5.3. [Ref. 18]



The new native Solaris version runs on Sun SPARCstations running Solaris 2.4/2.5 with OpenWindows 3.5, VADSself 6.2, and optionally TAE Plus 5.31.

Table 1 provides a listing of the approximate hard drive space required for the various software tools evaluated or actually used to build and run CAPS on Solaris 2.5. The totals were acquired using the UNIX disk usage command (`du -ks`).

<b>Hard Drive Space Requirements</b>	
Solaris 2.5 (full install)	400.0 MB
CAPS Release 2 (no source code)	47.4 MB
CAPS Release 2 (with source code)	150.3 MB
VADSself 6.2 Ada Compiler	52.0 MB
TAE Plus 5.31	48.0 MB
Synthesizer Generator 4.2	25.7 MB
Eli 4.4	14.8 MB
GNAT 3.05 Ada95 Compiler	14.8 MB
Motif 2.0 (source code)	119.0 MB

**Table 1. System Requirements for CAPS**

## **C. EVALUATION**

### **1. Solaris Compatibility Issues**

Being based on Unix SVR4, Solaris 2.5 is neither source nor binary compatible with SunOS 4.x. [Ref. 8] Therefore, applications developed for SunOS 4.x environment may not run correctly under Solaris 2.5 environment. Many of the commands in SunOS 4.x either

have different functions from the SunOS version or they no longer exist in Solaris. A listing of command changes can be found in the Solaris Transition Guide [Ref. 8], Appendix A, Command Reference Table.

To help with migration, Solaris 2.5 provides two optional compatibility packages: the SunOS/BSD Source Compatibility Package and the Binary Compatibility Package (BCP). The SunOS/BSD Source Compatibility Package contains a collection of SunOS 4.x and BSD commands, library routines, and header files not available with the Solaris 2.5 environment. The SunOS/BSD Source Compatibility Package is located in the `/usr/ucb` directory. To access the commands `/usr/ucb` should precede `/usr/bin` in your path. [Ref. 8]

The Binary Compatibility Package allows existing SunOS 4.x applications (linked statically and dynamically) to run under Solaris 2.5 without modification or recompilation. [Ref. 8] The Binary Compatibility Package consists of the Solaris 2.x and OpenWindows Binary Compatibility Packages which are optionally installed packages called `SUNWbcp` and `SUNWowbcp` respectively. More information can be found in the Binary Compatibility Guide. [Ref. 19]

It should be noted that these compatibility packages should serve only as a temporary fix, since such support will not always be there in future releases of Solaris. Therefore, all Solaris 2.5 applications should be developed using only the base environment. [Ref. 8]

## **2. X-Windows and Motif**

Solaris 2.5 includes the OpenWindows™ 3.5 windowing environment, which is based on X11 and Motif. Therefore, the Solaris 2.5 distribution comes with the X11R5 and Motif

1.2.4 runtime libraries included. The X11 libraries are located `/usr/openwin/lib` and the Motif libraries are located in `/usr/dt/lib`.

An early question in the research was whether or not separate installations of X-Windows and Motif were required to build and run CAPS. In the case of Motif, this is a considerable savings in cost if the runtime libraries could be used. Otherwise, the latest versions, X11R6 and Motif 2.0, would be installed. It was determined that the runtime libraries that are distributed with Solaris would be sufficient to run CAPS and TAE Plus 5.31, which also requires X11R5 and Motif 1.2.3 or higher. Additionally, as long as the include and library paths are properly setup during compilation, CAPS could be built dynamically using the included runtime libraries. However, this is not the case when compiling CAPS statically.

The executable files in the CAPS distribution are statically compiled to make them more portable. It was discovered while attempting to statically compile and load the graphical editor that the Motif library file `libXm.a` is no longer included with Solaris. Starting with Solaris 2.4, Sun stopped distributing the static Motif libraries in favor of the shared libraries. From Sun's bug report (Bug Id: 1195396) [Ref 20]:

The static version of the Motif library, `libXm.a` is no longer provided in Motif 1.2.3, which is being shipped as part of SDK for Solaris 2.4. The justification is that the shared (`.so`) runtime libraries are now bundled with the SunOS under `/usr/dt/lib`.

To solve this problem, an explicit installation of Motif is required. However, in order to build Motif from the source code, X11R5 and its source code must be present. Therefore,

future work on CAPS Release 2 and the planned Release 3 will require that X11R5 or X11R6/6.1 and Motif 2.0 be built from source. This was not accomplished at this time due to time constraints.

While X Windows is free, Motif must be purchased. Pre-compiled binaries can be purchased from third-party vendors for much less than the \$1000 educational site license for the Motif source code from OSF. Nevertheless, both X windows and Motif are industry standards and should be used.

### **3. TAE Plus**

TAE Plus 5.3l was installed using pre-compiled binaries for Solaris.2.x and Motif 1.2. TAE Plus also comes with Ada bindings which must be compiled separately with the user's Ada compiler [Ref. 21]. In the case of Solaris, the VADSself 6.2 Ada Compiler was used.

### **4. Synthesizer Generator and Eli**

The Synthesizer Generator and Eli were both built for Solaris 2.5 from the source code using the sun SPARCworks C/C++ Compiler (version 4.0) and the GNU C/C++ Compiler (GCC version 2.7.2). In general, greater success in compilation was achieved using the GNU C compiler than the SPARCworks one.

### **5. Compiling, Linking, and Loading**

Some general comments can be made regarding compiling software tools for the Solaris environment. The "tighter" compilers available for Solaris may complain when compiling anything other than strict ANSI C code, such as the older Kernighan and Ritchie (K&R) C code. In this case, the `-xs` compiler option for SPARCworks and the

-traditional compiler option for GCC should be tried. Additionally, whenever compiling for Solaris the `-lsocket` and `-lnsl` loader options must be included. If a compatibility package is required for compilation of SunOS 4.x source code, then `-L/usr/ucblib` and `-I/usr/ucbinclude` options will be needed. For more information on using the compilers with Solaris, see the man pages for `cc` (SPARCworks) and `gcc`.

## **6. Shell Scripts**

The major CAPS components are integrated using shell scripts. One of the major concerns of porting CAPS to Solaris was the compatibility of the shell scripts due to changes in the Unix commands from SunOS 4.3.x. Since the CAPS shell scripts use fairly basic shell commands, this proved not to be a problem. Other than changes made to accommodate changes in the Release 2 components, the only changes made due to Solaris were the locations of external components, such as TAE Plus and VADSself. The convention for Solaris is to place third-party software in the `/opt` directory.

Related to the shell scripts issue is the method used to display alerts generated from within the shell scripts. The original shell scripts used the `alert` command to display messages in a dialog box. This `alert` command was determined to be part of the InterViews package. Rather than rebuild InterViews on Solaris for just that one command, an alternative was found that provided the same capabilities and more. The alternative, `xmessage`, is a free X Windows contrib, which was compiled statically using GCC 2.7.2 and provided as part of the CAPS distribution. An alert script that calls `xmessage` with the appropriate options was

created in a manner such that it is used the same way as the `InterViews alert` command. Thus no changes were needed in any of the scripts that used the `alert` command.

A note should be made that the compilation of `xmessage` is not completely static. When compiling statically, a static link is forced for all libraries and the compiler searches only for the `.a` versions of the libraries. Unfortunately, `libdl.a`, which is required for `xmessage`, is unavailable. According to the SunSolve Symptoms and Resolution Database (document `srdb/2220`) [Ref. 22], the only solution is to compile `xmessage` statically except for `libdl` which is linked dynamically. The `ldd` command is very useful for determining and troubleshooting the dynamic dependencies of an executable file.



## IV. CONCLUSIONS AND FUTURE RESEARCH

### A. SUMMARY

This thesis research has laid the ground work for future development of CAPS for the Solaris operating system and its portability to other platforms and operating environments.

Table 2 provides a summary of accomplishments.

Task	Method
Installed Solaris 2.5 operating system	Installed from CDROM
Installed GNAT 3.05 Ada95 compiler and GNU 2.72 C/C++ compiler on Solaris 2.5	Installed by using pre-compiled binaries downloaded for Solaris 2.4
Installed TAE Plus on Solaris 2.5	Installed by using pre-compiled binaries for Solaris 2.x and Motif 1.2 Ada bindings compiled using VADSself 6.2 compiler
Installed Synthesizer Generator on Solaris 2.5	Built from source code using SPARCworks 4.0 and GNU 2.7.2 C/C++ compilers
Installed Eli on Solaris 2.5	
Compiled CAPS code on Solaris 2.5	Used C/C++ and Ada compilers for Solaris with Motif 1.2.4 and X11R5 runtime libraries

**Table 2. Summary of Accomplishments**

The Solaris 2.5 operating system and other programming tools were successfully installed and configured on four research workstations, creating a useable working



environment. This has also generated interest within the Computer Science Department to install Solaris on some of the departmental workstations.

The porting of CAPS to the Solaris 2.5 for Sun SPARCstations was a success; however, several problems were encountered that will make porting to other platforms difficult, but not impossible. These will require further research and I am confident they can be overcome.

The major problem in porting CAPS to the Solaris operating system was acquiring Solaris versions of the various tools needed to build CAPS. This problem is not limited to just porting the system to Solaris, but to any other operating system or platform.

Related to this problem was the difficulty compiling tools such as the Synthesizer Generator, Eli, and xmessage when the source code was available. Much of the older C code was not compatible with the new tighter Sun SPARCworks compiler. More success was achieved in this regard using the GNU C compiler. Additionally, compiler flags, includes, and library paths had to be modified to accommodate differences with Solaris over SunOS.

Another concern porting CAPS to Solaris was the compatibility of shell scripts due to changes in the Unix commands from SunOS 4.3.x. Some commands were changed, others were replaced with new commands, and still others were deleted with no SunOS equivalent provided. Since the CAPS shell scripts used fairly basic shell commands, this proved not to be a problem with the current release. However, researchers should be aware of this potential problem when developing future Solaris releases of CAPS with expanded capabilities and not rely on any compatibility packages.

## **B. SUGGESTIONS FOR FUTURE WORK**

CAPS is an evolving research project. As such there are ample opportunities for further work. As a result of this research, several areas have been identified for further work and improvement.

### **1. Alternatives to TAE Plus**

A major stumbling block to porting CAPS to other platforms is its reliance on TAE as the graphical user interface tool. TAE is only available on a limited number of platforms (see TAE Plus Supported Platforms at <http://www.cen.com/tae/taepf.html>) and is not free. Additionally, the TAE support for Ada was designed around SunAda 1.1 and is currently the only compiler that TAE directly supports. Use of another Ada compiler requires a significant porting effort for the Ada TAE bindings. Alternatives to TAE should be investigated and tried with CAPS. One promising possibility is Fresco, which is available for free from the X Consortium (<http://www.faslab.com/fresco/HowToGetIt.html>).

### **2. Ada 95 Compiler Integration**

Currently, CAPS and TAE produce Ada83 code and utilizes an Ada83 compiler. Work is currently underway to have CAPS produce Ada95 code and take advantage of the new features of the language. Research is needed to select one of the new Ada95 compilers that are due out during the last quarter of 1996 and first quarter 1997 (ObjectAda from Thomson Software and Spire from Rational Software Corporation) or the GNAT Ada95 compiler. However, to make a true Ada95 version of CAPS, the user interface tools must

be compatible with the Ada95 compiler, which is another problem with TAE. Also, work is needed to incorporate the new Ada95 compiler in the tool interface and CAPS setup scripts.

### **3. Tools Interface**

An improved user interface needs to be incorporated into CAPS. A class project for the Advanced Software Engineering class (CS4520) during the Winter Quarter 1996 at NPS produced such an improved interface that incorporates many of the tenets of good user interface design, plus utilizes the World Wide Web to add an extra dimension to the help facilities.

### **4. Personal Computer Version**

Further research is needed to port CAPS to the personal computer (PC) / Intel architecture. The Solaris operating system is already available for many platforms including the PC. Solaris x86 is the Solaris version for high end personal computers. Further research is needed to see if the port of CAPS to Solaris for Sun workstations can be transferred to a PC running Solaris x86. Another option is to port CAPS to a PC running the Linux version of Unix. Linux has the advantage of being freeware. Once again, the major obstacle will be the user interface tool (a version of TAE is not available for the Intel architectures) and the availability of a compatible Ada83/Ada95 compiler. It should be possible to build x86 and Linux versions of Eli and the Synthesizer Generator from the source code.

### **5. Shell Scripts**

CAPS is a collection of software modules which communicate with each other through the use of shell scripts. As the complexity of CAPS grows, a more powerful scripting

language may be required. A possible candidate is the Perl Language, which is also available for free (<http://www.perl.com/perl/info/software.html>). Alternatively, the scripts could be eliminated by placing the system commands in an Ada wrapper. When porting to another system, only the contents of the system calls in the Ada modules would have to be changed.

## **6. Alert Dialog Boxes**

The CAPS Release 1 relied on the alert command which was provided as part of the Interviews package. The alert command was used in the CAPS shell scripts to produce a dialog box with an alert message informing the user that a task was completed successfully. For Release 2, xmessage, a free X-Windows contrib, was statically compiled and used instead of the Interviews alert. However, a better approach should be taken. There is already an alert dialog box that was created with TAE within CAPS. The alert messages using xmessage from the shell scripts should be incorporated into the Ada code using the CAPS alert dialog box. This would provide a uniform look for all of the CAPS dialog boxes and would reduce the reliance on outside programs.

## **7. New Environments**

In the ever changing world of computers and software, new operating environments are already appearing on the horizon which will need investigating. Such include a new versions of Solaris (2.5.1), the Common Desktop Environment (CDE), and the latest evolution of the X Window System, code named Broadway, which will offer the ability to access and execute applications over the web.

## **C. CONCLUSION**

If the intent is to continue distributing CAPS freely and to increase its usage by the software engineering community, then every effort should be made to port CAPS to a wide variety of platforms and to use tools that are either freeware (i.e., Fresco and X-Windows) or are as low cost to the consumer as possible.

## LIST OF REFERENCES

1. Luqi, "Computer-Aided Software Prototyping," *IEEE Computer*, Vol 24, No. 9, September 1991.
2. Luqi, V. Berzins, and R. Yeh, "A Prototyping Language for Real-Time Software," *IEEE Transactions on Software Engineering*, October 1988.
3. Cooke, Robert P., *Technology Transfer of the Computer-Aided Prototyping System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1996.
4. Luqi, "Software Evolution Through Rapid Prototyping," *IEEE Computer*, May 1989.
5. Luqi and M. Shing, "Teaching Hard Real-Time Software Development via Prototyping," *Proceeding of the ACM/IEEE International Workshop on Software Engineering Education*, May 1994.
6. Luqi, "Summary - What CAPS Will Do For Its Users", forthcoming NPS Technical Report, Computer Science Department, Naval Postgraduate School, 1996.
7. Sun Microsystems, *Solaris Network Operating Environment*, White Paper, Sun Microsystems, 1995.
8. Sun Microsystems, *Solaris 1.x to Solaris 2.x Transition Guide*, Sun Microsystems, November 1995.
9. Sun Microsystems, *System Administration Guide, Volume II*, Sun Microsystems, November 1995.
10. Sun Microsystems, *Solaris 2.5: The Gateway to Multiarchitecture Applications*, available from [http:// www.sun.com/solaris/2.5/multi-arch.html](http://www.sun.com/solaris/2.5/multi-arch.html).
11. Gildea, S., *X Window System, Version 11, Release 6 Release Notes*, X Consortium, May 16, 1994.
12. Open Software Foundation, *The New User Interface Hybrids: Integrating User Environments with OSF/Motif*, OSF White Paper, March, 1994.

13. Open Software Foundation, *OSF/Motif 2.0 Data Sheet*, OSF-M-DS-694-1, Open Software Foundation, Inc., June 1994.
14. Century Computing, *TAE Plus Overview*, Century Computing, Inc., September, 1993.
15. Century Computing, *Release Notes for TAE Plus Version 5.3l*, Century Computing, Inc., May, 1995.
16. GrammaTech, *The Synthesizer Generator Reference Manual*, Release 4.2, GrammaTech, Inc., Ithaca, New York, 1995.
17. *Eli Overview*, available from <http://www.cs.colorado.edu/~eliuser/ExplainEli.html>.
18. Brockett, Jim, *The Computer-Aided Prototyping System (CAPS): Installation Manual*, CAPS Release 1, Naval Postgraduate School, Monterey, California, October, 1994.
19. Sun Microsystems, *Binary Compatibility Guide*, Sun Microsystems, November 1995.
20. Sun Microsystems, Sun Bug Report (Bug Id: 1195396), SunSolve Online, <http://192.9.9.24/>.
21. Sun Microsystems, SunSolve Document srdb/2220, SunSolve Online, <http://192.9.9.24/>.
22. Century Computing, *TAE Plus Ada Reference Manual*, Century Computing, Inc., September, 1993.

## BIBLIOGRAPHY

Anderson, G. and P. Anderson, *The UNIX™ C Shell Field Guide*, Prentice Hall, 1986.

Berzins, V. and Luqi, "Rapidly Prototyping Real-Time Systems," *IEEE Software*, September 1988.

Berzins, V. and Luqi, *Software with Abstractions*, Addison-Wesley, 1991.

Boehm, B. W., T. E. Gray, and T. Seewaldt, "Prototyping Versus Specifying: A Multiproject Experiment," *IEEE Transactions on Software Engineering*, May 1984.

Brockett, Jim, *The Computer-Aided Prototyping System (CAPS): A CAPS Tutorial*, CAPS Release 1, Naval Postgraduate School, Monterey, California, October, 1994.

Century Computing, *TAE Plus Ada Reference Manual*, Century Computing, Inc., September, 1993.

Century Computing, *TAE Plus Overview*, Century Computing, Inc., September, 1993.

Century Computing, *TAE Plus Programmer's Manual*, Century Computing, Inc., September, 1993.

Century Computing, *TAE Plus System Manager's Guide*, Century Computing, Inc., September, 1993.

Century Computing, *TAE Plus User Interface Developer's Guide*, Century Computing, Inc., September, 1993.

Cummings, Mary Ann, *The Development of User Interface Tools for the Computer-Aided Prototyping System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1990.

Dixon, Robert Mobley, *The Design and Implementation of a User Interface for the Computer-Aided Prototyping System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1992.

Dougherty, D., *sed & awk*, O'Reilly & Associates, 1993.



GrammaTech, *The Synthesizer Generator Reference Manual*, Release 4.2, GrammaTech, Inc., Ithaca, New York, 1995.

Grosenheider, Scott Robert, *Enhancements for the CAPS Prototyping System Description Language Syntax-Directed Editor*, M.S. Thesis, Naval Postgraduate School, Monterey, California, March 1996.

Luqi, "Software Evolution Through Rapid Prototyping," *IEEE Computer*, May 1989.

Luqi, "A Graph Model for Software Evolution," *IEEE Transactions on Software Engineering*, August 1990.

Luqi, "Computer-Aided Software Prototyping," *IEEE Computer*, Vol 24, No. 9, September 1991.

Luqi, "Computer-Aided Prototyping for a Command-and-Control System Using CAPS," *IEEE Software*, January 1992.

Luqi and M. Ketabchi, "A Computer Aided Prototyping System," *IEEE Software*, March 1988.

Luqi, V. Berzins, and R. Yeh, "A Prototyping Language for Real-Time Software," *IEEE Transactions on Software Engineering*, October 1988.

Luqi and W. Royce, "Status Report: Computer-Aided Prototyping," *IEEE Software*, November 1991.

Luqi and M. Shing, "Teaching Hard Real-Time Software Development via Prototyping," *Proceeding of the ACM/IEEE International Workshop on Software Engineering Education*, May 1994.

Luqi and M. Shing, "Real-Time Scheduling for Software Prototyping," *Journal of Systems Integration*, Vol. 6, Nos. 1/2, March 1996.

Ozdemir, Dogan, *The Design and Implementation of a Reusable Component Library and a Retrieval/Integration System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1992.

Raum, Henry G., *Design and Implementation of an Expert User Interface for the Computer-Aided Prototyping System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1988.

Reps, T. and T. Teitelbaum, *The Synthesizer Generator: A System for Constructing Language-Based Editors*, Springer-Verlag, New York, 1989.

Schneiderman, B., *Designing the User Interface*, Addison-Wesley, 1987.

Sun Microsystems, *Binary Compatibility Guide*, Sun Microsystems, November 1995.

Sun Microsystems, *Solaris 1.x to Solaris 2.x Transition Guide*, Sun Microsystems, November 1995.

Sun Microsystems, *System Administration Guide, Volume I*, Sun Microsystems, November 1995.

Sun Microsystems, *System Administration Guide, Volume II*, Sun Microsystems, November 1995.

SunSoft, *Solaris Porting Guide*, Prentice Hall, 1995.

Winsor, J., *Solaris System Administrator's Guide*, SunSoft Press/Ziff-Davis Press, 1993.

Winsor, J., *Solaris Advanced System Administrator's Guide*, SunSoft Press/Ziff-Davis Press, 1993.



## APPENDIX A. LOCAL SOLARIS SYSTEM INFORMATION

The following is a selected collection of notes on the installation and general system administration of Solaris 2.5 on Naval Postgraduate School Computer Science Department workstations (*perseus*, *aldebaran*, *suns8*, and *suns9*):

### LOCAL INSTALLATION NOTES

Determine IP address from a departmental system:

```
ypcat hosts | grep <host name>
```

Add CDROM drive (if not installed):

```
halt
turn system off
install cdrom
turn system on
boot -r          (to ensure system recognizes new devices)
probe-scsi       (check devices recognized by the system)
```

Start installation:

```
boot cdrom
or
boot sd(0,6,2) for aldebaran
```

Domain Name Server:

```
compsci.nps.navy.mil
```

Subnet:

```
255.255.255.0
```

Possible Customizations:

```
select separate partition for /opt
rename /opt → /var
rename /export/home → /opt
```

Immediately after installation set up a swap file:

Note: Total swap space should be 2 times the RAM.

To prepare an empty partition for a swapfile:

```
df -k
umount /<partition>
newfs -m 3 /dev/dsk/... NOTE: only do this on an empty partition!
mount -a
```

Create and add a swapfile:

```
mkfile 64m /export/home/swapfile1 (or /opt/swapfile1)
    ^Size of file.
swap -l
swap -s
swap -a /<partiton>/swapfile1
swap -l
swap -s
Add swap file to /etc/vfstab file
```

Modify /etc/passwd:

for root change /sbin/sh to /usr/bin/csh

Modify /etc/hosts:

copy /etc/hosts from libra as /etc/hosts-master  
append libra's /etc/hosts information to /etc/passwd and remove redundant entries.

To be able to handle different shells, create /etc/shells containing the following:

```
/bin/csh
/bin/sh
/usr/local/bin/tcsh
```

Default Computer Science Department Files:

```
/users/work1/default/
```

Determine path of Open Windows:

```
which openwin
```

Start Open Windows:

```
/usr/openwin/bin/openwin
```

To eject CDROM

```
eject
```

## SYSTEM INFORMATION

### **perseus:**

131.120.1.38  
64 MB RAM  
424MB Internal  
/  
/usr  
/var  
424MB Internal  
/opt  
1200MB External  
/work  
/usr/local  
Remote Mounted  
/local/lang  
8mm Tape Backup

### **aldebaran:**

131.120.1.39  
64 MB RAM  
424MB Internal  
1600MB External

## SOLARIS SYSTEM ADMINISTRATION NOTES

To add/remove an external device:

```
halt
turn system off
install device      (ie. CDROM)
turn system on
boot -r             (to ensure system recognizes new devices)
probe-scsi          (check devices recognized by the system)
```

To remove PROM password:

```
setenv security-mode
reset
```

Rebooting as a single user:

```
b sd(0,0,0)vmunix -s
csh
setenv TERM sun      now you can use vi to make changes to bring system
```

Starting/Stopping NFS:

```
sh /etc/init.d/nfs.client stop
sh /etc/init.d/nfs.client start

sh /etc/init.d/nfs.server stop
sh /etc/init.d/nfs.server start
```

Stopping the system if it freezes during boot-up:

```
press STOP/L1 and "a" simultaneously
halt
boot as a single user
```

Various Commands:

b sd(0,0,0)vmunix -s	reboot as single user
catman -w	creates "windex" to be used by man -k
dmesg	show hardware found on system
find . -name <name> -print	finds filename from current directory and displays its location
nslookup <host name>	
pkginfo	displays installed packages
ping <host name>	
reboot	reboots system
suninstall	run install program if already in memory

Various Files:

<code>/.rhosts</code>	
<code>/etc/dfs/dfstab</code>	edit this file to export files to other systems
<code>/etc/hosts</code>	
<code>/etc/hosts.equiv</code>	
<code>/etc/nsswitch.conf</code>	
<code>/etc/passwd</code>	contains user information
<code>/etc/printcap</code>	contains printer information
<code>/etc/resolv.conf</code>	
<code>/etc/vfstab</code>	contains list of remote disks to mount use <code>mkdir &lt;directory name&gt;</code> to create the mount points

#### T Shell (tcsh) Commands:

<code>h</code>	shows history of commands
<code>!<code>&lt;command #&gt;</code></code>	redoes command in history file
<code>^&lt;string1&gt;^&lt;string2&gt;</code>	replaces string1 in previous command with string2

#### To determine a user's id #:

```
ypcat passwd | grep <user name>
```

#### Admin Tool:

```
admintool&
```

#### Creating User Accounts:

```
use admintool
or
cut & paste users from /etc/passwd on another machine to the new /etc/passwd file
passwd <username>
pwconv
```

**Note:** For Solaris, `su <username>` does not cause the environment to change. You need to manually switch to the new home directory and source the `.cshrc` file. Alternatively, use `su -` to get new user's environment.



### Example Full Backup Script:

```
#!/bin/csh -f

echo "rewinding"
mt -f /dev/rmt/0 rewind

fsck /
fsck /var
fsck /usr
fsck /opt
fsck /work
fsck /usr/local

echo "dumping root"
ufsdump 0ufbsd /dev/rmt/0n 126 6000 54000 /

echo "dumping /var"
ufsdump 0ufbsd /dev/rmt/0n 126 6000 54000 /var

echo "dumping /usr"
ufsdump 0ufbsd /dev/rmt/0n 126 6000 54000 /usr

echo "dumping /opt"
ufsdump 0ufbsd /dev/rmt/0n 126 6000 54000 /opt

echo "dumping /work"
ufsdump 0ufbsd /dev/rmt/0n 126 6000 54000 /work

echo "dumping /usr/local"
ufsdump 0ufbsd /dev/rmt/0n 126 6000 54000 /usr/local

echo "rewinding"
mt -f /dev/rmt/0 rewind
```

### **Example Incremental Backup Script:**

```
#!/bin/csh -f

echo "rewinding"
mt -f /dev/rmt/0 rewind

fsck /
fsck /var
fsck /usr
fsck /opt
fsck /work
fsck /usr/local

echo "dumping root"
ufsdump 1ufbsd /dev/rmt/0n 126 6000 54000 /

echo "dumping /var"
ufsdump 1ufbsd /dev/rmt/0n 126 6000 54000 /var

echo "dumping /usr"
ufsdump 1ufbsd /dev/rmt/0n 126 6000 54000 /usr

echo "dumping /opt"
ufsdump 1ufbsd /dev/rmt/0n 126 6000 54000 /opt

echo "dumping /work"
ufsdump 1ufbsd /dev/rmt/0n 126 6000 54000 /work

echo "dumping /usr/local"
ufsdump 1ufbsd /dev/rmt/0n 126 6000 54000 /usr/local

echo "rewinding"
mt -f /dev/rmt/0 rewind
```

### **Remote Backups:**

Make changes similar to the following:

```
echo "rewinding"
rsh perseus mt -f /dev/rmt/0 rewind

echo "dumping root"
ufsdump 0ufbsd perseus:/dev/rmt/0n 126 6000 54000 /
```

### **Restoring or Checking Tape Files:**

```
ufsrestore -ibf 126 /dev/rmt/0
```

### **Advancing to Next Tape Block:**

```
mt -f /dev/rmt/0n fsf 1
```

### **Mounting a Local Disk:**

```
mount /dev/dsk/c0t1d0s0 /mnt
```

### **Mounting a NFS File System From Another Host:**

```
mount -F nfs -o suns8:/opt /opt
```

For this to work, the other host must “share” the file (see below).

### **Sharing a NFS File system With Other Hosts:**

Use the following command at the command prompt or in the `/etc/dfs/dfstab` file for automatic sharing at boot-up:

```
share -F nfs -o rw=perseus /opt
```

To allow another host to be root, use:

```
share -F nfs -o rw=perseus, root=pegasus /opt
```

### **Unmounting a File System:**

```
umount /mnt
```

Note - the partition to be unmounted cannot be in use (part of the current directory's path).

### **Setting X-Windows Environment:**

```
set path=($path ...)
set path = (/opt/local/X11R6/bin $path)
which xinit
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/opt/local/X11R6/lib

xinit -- /usr/openwin/bin/Xsun
```

### Miscellaneous Commands:

env	displays current environment
/usr/openwin/bin/kbd_mode -a	resets keyboard
rcp	remote copy
rsh	remote shell command
umask 022	
umask 077	
xwd   xpr -device ps   lpr	captures and prints an X-Window

### View Dynamic Libraries of an Executable:

ldd <filename>

## Formatting and Mounting an Optical Disk:

Determine device name of optical disk by typing `dmesg` and reviewing the boot messages. On wonton it is `sd3` and on sun60 it is `sd2`.

- Run formatting program

```
format sd2
select type as MaxOptixTahiti 2
```
- Label each side (no need to format)
- Create file system (caution: destroys any old data on disk!)

```
newfs -c 32 -i 16384 -t 8 -C 6 /dev/rsd2c
```
- Mount optical disk

```
mount /dev/sd2c /optic
```

## Floppy Disks:

Solaris automounts Unix and DOS floppies when detected. Solaris can also format, read, and write DOS floppies using standard Unix commands without using external software such as `mttools`.

<code>volcheck</code>	causes Solaris to check for a floppy
<code>cd /floppy</code>	change directory to the floppy
<code>cd</code>	exit floppy's directory before ejecting
<code>eject floppy</code>	eject the floppy from the drive

If needed, the usual device names for the floppy drive are:

```
/dev/rfd0c
/dev/fd0c
```

## Environment Setup:

The following .cshrc file was used on suns8 to run the Solaris version of CAPS:

```
#!/bin/csh

# CAPS LAB SOLARIS 2.5 SYSTEM ONLY .CSHRC FILE

# Set echo below inorder to echo each command as it is executed.
# Useful for troubleshooting.
# set echo

setenv LM_LICENSE_FILE
/local/license/license.dat:/local/license/license.dat.scholarpak

set path = (/opt/local/bin /opt/cygnus/bin \
            /bin /usr/bin /usr/ccs/bin /opt/X11R5/bin /usr/sbin \
            /etc /usr/etc /usr/share/lib /opt/java/bin \
            /local/lang /usr/local/bin /usr/openwin/bin /usr/ucb . )

setenv LD_LIBRARY_PATH /usr/openwin/lib\:/usr/ucblib\:/usr/lib
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/usr/X11R5/lib\:/opt/local/lib
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/opt/cygnus/lib
setenv MANPATH /usr/share/man\:/opt/local/man\:/opt/X11R5/man
setenv MANPATH $MANPATH\:/opt/ICS/Motif1.2.4/usr/man:/usr/man:
setenv MANPATH $MANPATH\:/usr/openwin/man\:/opt/cygnus/man

set noclobber
set ignoreeof
limit coredumpsize 0
umask 077

# skip remaining setup if not an interactive shell
if ($?USER == 0 || $?prompt == 0) exit

setenv OPENWINHOME /usr/openwin
setenv AB_CARDCATALOG /opt/SUNWabe/ab_cardcatalog
alias answerbook /usr/openwin/bin/answerbook
setenv EDITOR /usr/bin/vi
setenv CLASSPATH .\:/opt/java/lib/classes.zip

# environment stuff so we look like BSD unix
set history=40

alias cd          'cd \!*;echo $cwd'
alias cp          'cp -i'
alias mv          'mv -i'
alias pwd         'echo $cwd'
#alias rm         'rm -i'
alias print       'enscript -G -2r'

alias h           history
alias ls          'ls -p'
alias ll          'ls -al'
```

```

alias la          'ls -a'
alias lo          logout
alias m           'more'
alias mail        'mail -hr'
alias man         'man -F'

alias ^L          clear
alias .           'echo $cwd'
alias ..          'set dot=$cwd;cd ..'
alias ,           'cd $dot'
alias open        'chmod go+r'
alias shut        'chmod go-r'
alias bye         logout

#set up for which csh
#set prompt based on shell type
switch($shell:t)
  case csh:
    set prompt = "`hostname | sed s/IS//`: "
    breaksw
  case tcsh:
    set prompt="%S%m:%/>>%s "
    breaksw
  default:
    echo "Don't know which shell, you get default prompt"
endsw

#software environment stuff
setenv EMACS solemacs
setenv XDEVICE /dev/fbs/cgsix0

# set up for X11R6
if (-d /opt/X11R6) then
  #set path = (/opt/X11R6/bin $path)
  #setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/opt/X11R6/lib
endif

# set up for GNAT Ada95 Compiler for Solaris
if (-d /opt/gnu) then
  set path = (/opt/gnu/bin $path)
  #alias usegnat 'source /usr/local/gnat/usegnat'
endif

# FMHOME line added by the FrameMaker setup program
if ($?TERM) then
  switch ($TERM)
    case sun-cmd:
      setenv FMHOME /opt/frame; set path=($path $FMHOME/bin)
      breaksw;
    case xterm:
      setenv FMHOME /opt/frame; set path=($path $FMHOME/bin)
      breaksw;
  endsw
else
  alias maker 'echo "frame not supported for this TERM"'

```

```

endif

# set up for Vads self compilers
if(-d /opt/vads) then
    set path = ($path /opt/vads/bin)
    setenv LM_LICENSE_FILE
/local/license/licenses_combined\:/local/license/license.dat.scholarpak
    setenv MANPATH $MANPATH\:/opt/vads/man
    #alias a.mklib a.mklib -f . /opt/vads/self/standard
endif

# setup for local sun software
if(-d /opt/SUNWspro) then
    set path = (/opt/SUNWspro/SC4.0/bin $path)
    setenv LD_LIBRARY_PATH /opt/SUNWspro/SW3.1/lib\:$LD_LIBRARY_PATH
    setenv LD_LIBRARY_PATH /opt/SUNWspro/SC4.0/lib\:$LD_LIBRARY_PATH
    setenv MANPATH /local/lang/man\:$MANPATH
endif

# setup up for campus sun software
# this works with both SUNOS and SOLARIS
if(-d /local/lang) then
    set path= ($path /local/lang/bin)
    setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/local/lang/SC3.0.1/lib
    setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/local/lang/SW3.0.1/lib
    setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/local/lang/lib
    setenv MANPATH $MANPATH\:/local/lang/man
endif

# set up for TAE Plus 5.31 for Solaris
if (-d /opt/tae5.31) then
    setenv TAE /opt/tae5.31
    source $TAE/bin/csh/taesetup
    set path =($path $TAE$BIN $TAE$DEMO$BIN)
    setenv MANPATH $MANPATH\:$TAE$MAN\:$TAE/Xtae/man
    # setup for TAE with ada bindings

    # setup for Motif under Solaris
    setenv OPENWINHOME /usr/openwin
    setenv MOTIFHOME /usr/dt

    setenv LD_LIBRARY_PATH $MOTIFHOME/lib:$OPENWINHOME/lib
    setenv UIDPATH %U:/opt/SUNWmfdm/lib/uid/%U
    setenv XFILESEARCHPATH $MOTIFHOME/lib/%T/%N%S
    setenv XMBINDDIR $MOTIFHOME/lib/bindings

    set path=($OPENWINHOME/bin $path)
    set path=($path $MOTIFHOME/bin) # for uil and xmbind
    set path=($path /opt/SUNWmfwm/bin) # for mwm

    setenv MANPATH ${MANPATH}:${OPENWINHOME}/man
    setenv MANPATH ${MANPATH}:${MOTIFHOME}/man
    setenv MANPATH ${MANPATH}:/opt/SUNWmfwm/man
endif

```



```
# setup for CAPS (Solaris)
setenv CAPSHOME /export/home/irwin/CAPS.2.SOLARIS.TEST
set path =($path $CAPSHOME/bin)
source $CAPSHOME/bin/CAPSsetup
```

```
# set up for Grasp
if (-d /opt/graspada) then
    setenv GRASP_HOME /opt/graspada
    set path = ($path $GRASP_HOME/bin)
    alias grasp grasp.static
endif
```

```
echo "                WELCOME to SOLARIS"
```

## Initial .cshrc File:

```
# @(#)cshrc 1.11 89/11/29 SMI

# set up for SOLARIS 2.5 system only, not SUN OS

set path = (/opt/local/bin \
            /bin /usr/bin /usr/ccs/bin /usr/sbin \
            /etc /usr/etc /usr/share/lib /local/lang /usr/local/bin \
            /usr/openwin/bin /usr/ucb . )

# set default man and library paths
setenv MANPATH /usr/man\:/opt/local/man:/local/shareware/unix/man
setenv LD_LIBRARY_PATH /usr/lib\:/usr/local/lib:/opt/local/lib

umask 077
set noclobber
limit coredumpsize 0

# skip remaining setup if not an interactive shell
if ($?USER == 0 || $?prompt == 0) exit

set history=40
alias h history
alias man 'man -F'
alias mail 'mail -hr'

setenv OPENWINHOME /usr/openwin
setenv AB_CARDCATALOG /opt/SUNWabe/ab_cardcatalog
alias answerbook /usr/openwin/bin/answerbook

# setup for X-Emacs for Solaris
#setenv EMACS solemacs
#setenv XDEVICE /dev/fbs/cgsix0

# set up for X11R6 for Solaris
#if (-d /opt/X11R6) then
#   set path = (/opt/X11R6/bin $path)
#   setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/opt/X11R6/lib
#   setenv MANPATH $MANPATH\:/opt/X11R6/man
#endif

# set up for GNAT 3.03 Ada Compiler for Solaris
if (-d /opt/gnu) then
    set path = (/opt/gnu/bin $path)
endif

# set up for TAE Plus 5.31 for Solaris
if (-d /opt/tae5.31) then
    setenv TAE /opt/tae5.31
    source $TAE/bin/csh/taesetup
    set path =($path $TAEBIN $TAEDEMOBIN)
    setenv MANPATH $MANPATH\:$TAEMAN\:$TAE/Xtae/man
    # setup for TAE with ada bindings
```

```
endif
```

```
#Setup for CAPS
```

```
#setenv CAPSHOME /export/home/irwin/CAPS.RELEASE.1
```

```
#source $CAPSHOME/bin/CAPSsetup
```

## Default .cshrc File:

```
#!/bin/csh

# CAPS LAB SOLARIS 2.5 SYSTEM ONLY .CSHRC FILE

# Set echo below inorder to echo each command as it is executed.
# Useful for troubleshooting.
# set echo

setenv LM_LICENSE_FILE
/local/license/license.dat:/local/license/license.dat.scholarpak

set path = (/opt/local/bin /opt/cygnus/bin \
            /bin /usr/bin /usr/ccs/bin /opt/X11R5/bin /usr/sbin \
            /etc /usr/etc /usr/share/lib /opt/java/bin \
            /local/lang /usr/local/bin /usr/openwin/bin /usr/ucb . )

setenv LD_LIBRARY_PATH /usr/openwin/lib:/usr/ucblib:/usr/lib
setenv LD_LIBRARY_PATH
$LD_LIBRARY_PATH:/usr/X11R5/lib:/opt/local/lib
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/opt/cygnus/lib
setenv MANPATH /usr/share/man:/opt/local/man:/opt/X11R5/man
setenv MANPATH $MANPATH:/opt/ICS/Motif1.2.4/usr/man:/usr/man:
setenv MANPATH $MANPATH:/usr/openwin/man:/opt/cygnus/man

set noclobber
set ignoreeof
limit coredumpsize 0
umask 077

# skip remaining setup if not an interactive shell
if ($?USER == 0 || $?prompt == 0) exit

setenv OPENWINHOME /usr/openwin
setenv AB_CARDCATALOG /opt/SUNWabe/ab_cardcatalog
alias answerbook /usr/openwin/bin/answerbook
setenv EDITOR /usr/bin/vi
setenv CLASSPATH ./:/opt/java/lib/classes.zip

# environment stuff so we look like BSD unix
set history=40

alias cd          'cd \!*;echo $cwd'
alias cp          'cp -i'
alias mv          'mv -i'
alias pwd         'echo $cwd'
alias rm          'rm -i'
#alias print      'enscript -G -2r'

alias h           history
alias ls          'ls -p'
alias ll          'ls -al'
alias la          'ls -a'
```

```

alias lo          logout
alias m           'more'
alias mail        'mail -hr'
alias man         'man -F'

alias ^L          clear
alias .           'echo $cwd'
alias ..          'set dot=$cwd;cd ..'
alias ,           'cd $dot'
alias open        'chmod go+r'
alias shut        'chmod go-r'
alias bye         logout

#set up for which csh
#set prompt based on shell type
switch($shell:t)
  case csh:
    set prompt = "`hostname | sed s/IS//`: "
    breaksw
  case tcsh:
    set prompt="%S%m:%/>>%s "
    breaksw
  default:
    echo "Don't know which shell, you get default prompt"
endsw

#software environment stuff
setenv EMACS solemacs
setenv XDEVICE /dev/fbs/cgsix0

# set up for X11R6
if (-d /opt/X11R6) then
  #set path = (/opt/X11R6/bin $path)
  #setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/opt/X11R6/lib
endif

# set up for GNAT Ada95 Compiler for Solaris
if (-d /opt/gnu) then
  set path = (/opt/gnu/bin $path)
  #alias usegnat 'source /usr/local/gnat/usegnat'
endif

# FMHOME line added by the FrameMaker setup program
if ($?TERM) then
  switch ($TERM)
    case sun-cmd:
      setenv FMHOME /opt/frame; set path=($path $FMHOME/bin)
      breaksw;
    case xterm:
      setenv FMHOME /opt/frame; set path=($path $FMHOME/bin)
      breaksw;
  endsw
else
  alias maker 'echo "frame not supported for this TERM"'
endif

```

```

# set up for Vads self compilers
if(-d /opt/vads) then
    set path = ($path /opt/vads/bin)
    setenv LM_LICENSE_FILE
/local/license/licenses_combined\:/local/license/license.dat.scholarpak
    setenv MANPATH $MANPATH\:/opt/vads/man
    #alias a.mklib a.mklib -f . /opt/vads/self/standard
endif

# setup for local sun software
if(-d /opt/SUNWspro) then
    set path = (/opt/SUNWspro/SC4.0/bin $path)
    setenv LD_LIBRARY_PATH /opt/SUNWspro/SW3.1/lib:$LD_LIBRARY_PATH
    setenv LD_LIBRARY_PATH /opt/SUNWspro/SC4.0/lib:$LD_LIBRARY_PATH
    setenv MANPATH /local/lang/man:$MANPATH
endif

# setup up for campus sun software
# this works with both SUNOS and SOLARIS
if(-d /local/lang) then
    set path= ($path /local/lang/bin)
    setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/local/lang/SC3.0.1/lib
    setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/local/lang/SW3.0.1/lib
    setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH\:/local/lang/lib
    setenv MANPATH $MANPATH\:/local/lang/man
endif

# set up for TAE Plus 5.31 for Solaris
if (-d /opt/tae5.31) then
    setenv TAE /opt/tae5.31
    source $TAE/bin/csh/taesetup
    set path =($path $TAEBIN $TAEDEMOBIN)
    setenv MANPATH $MANPATH\:$TAEMAN\:$TAE/Xtae/man
    # setup for TAE with ada bindings
endif

#Setup for CAPS
#setenv CAPSHOME /export/home/irwin/CAPS.RELEASE.1
#source $CAPSHOME/bin/CAPSsetup

echo "                WELCOME to SOLARIS"

```

### Example /etc/dfs/dfstab From *suns8*:

```
# place share(1M) commands here for automatic execution
# on entering init state 3.
#
# share [-F fstype] [ -o options] [-d "<text>"] <pathname> [resource]
# .e.g,
# share -F nfs -o rw=engineering -d "home dirs" /export/home2

share -F nfs -o ro=suns9:perseus:aldebaran /opt
share -F nfs -o rw=suns9:perseus:aldebaran /work1
share -F nfs -o rw=suns9:perseus:aldebaran /work2
share -F nfs -o rw=suns9:perseus:aldebaran /work3
share -F nfs -o rw=suns9:perseus:aldebaran /work4
```

### Example /etc/vfstab From *suns8*:

# device # to mount #	device to fsck	mount point	FS type	fsck pass	mount at boot	mount options
#/dev/dsk/c1d0s2	/dev/rdsk/c1d0s2	/usr	fs	1	yes	-
fd	-	/dev/fd	fd	-	no	-
/proc	-	/proc	proc	-	no	-
/dev/dsk/c0t3d0s1	-	-	swap	-	no	-
/dev/dsk/c0t3d0s0	/dev/rdsk/c0t3d0s0	/	ufs	1	no	-
/dev/dsk/c0t3d0s6	/dev/rdsk/c0t3d0s6	/usr	ufs	1	no	-
/dev/dsk/c0t3d0s3	/dev/rdsk/c0t3d0s3	/var	ufs	1	no	-
/dev/dsk/c0t1d0s7	/dev/rdsk/c0t1d0s7	/export/home	ufs	2	yes	-
/dev/dsk/c0t3d0s5	/dev/rdsk/c0t3d0s5	/opt	ufs	2	yes	-
swap	-	/tmp	tmpfs	-	yes	-
/export/home/swapfile1	-	-	swap	-	no	-
/dev/dsk/c1t1d0s6	/dev/rdsk/c1t1d0s6	/work1	ufs	2	yes	-
/dev/dsk/c1t3d0s6	/dev/rdsk/c1t3d0s6	/work2	ufs	2	yes	-
/dev/dsk/c1t0d0s6	/dev/rdsk/c1t0d0s6	/work3	ufs	2	yes	-
/dev/dsk/c1t2d0s6	/dev/rdsk/c1t2d0s6	/work4	ufs	2	yes	-

### Example /etc/auto\_direct From *suns8*:

# Key	[Mount-options]	locations
# first add path stuff		
/var/mail	-rw,intr,soft	taurus:/var/spool/mail
/opt/vads	-ro,intr,soft	virgo:/lang/solaris/vads6.2.3
/opt/SUNWspro		virgo:/lang/solaris/SUNWspro
/opt/frame		virgo:/opt/local/frame
/opt/java		virgo:/opt/local/java
/opt/cygnus		virgo:/opt/cygnus
/export/home/berzins	-rw,intr,soft	suns9:/export/home/berzins
/n/sun51/work	-rw,intr,soft	sun51:/work
/n/sun52/work	-rw,intr,soft	sun52:/work
/n/sun53/work	-rw,intr,soft	sun53:/work
/n/sun54/work	-rw,intr,soft	sun54:/work
/n/sun55/work	-rw,intr,soft	sun55:/work
/n/sun57/work	-rw,intr,soft	sun57:/work
/n/suns5/work	-rw,intr,soft	suns5-caps:/work
/n/suns7/work	-rw,intr,soft	suns7-caps:/work

### Example /etc/auto\_home From *suns8*:

```
# Home directory map for automounter
#
+auto_home
```

### Example /etc/auto\_local From *suns8*:

#	
#license	libra:/usr/local/license
license	virgo:/sunos/license
help	virgo:/sunos/doc
lang	virgo:/lang/solaris/SUNWspro

### Example /etc/auto\_master From *suns8*:

# Master map for automounter		
#		
/-	auto_direct	-ro,intr,soft
/local	auto_local	-ro,intr,soft
/users	auto_home	-rw,intr,soft





## APPENDIX B. SHELL SCRIPTS

This appendix contains the CAPS Release 2 shell scripts (presented in alphabetical order). The shell scripts are used to integrate the individual CAPS modules. The scripts will work unmodified on both Solaris and SunOS systems. The CAPSsetup script may need editing to tailor CAPS to a particular system's environment.

<b>CAPS Release 2 Scripts:</b>
alert
browse_sb.script
caps
caps_m.exe
CAPSsetup
compile.script
edit_ada.script
edit_interface.script
edit_psd1.script
execute.script
make.script
save_ada_files
show_ops.exe
tae_to_caps
temp_dir.script
translate.script

## alert

```
#####
# module: alert
# Description:
#   This script is used to pass an alert message to xmessage with the
#   proper formatting.
# Usage:
#   echo <text> | alert
# Outputs:
#   Displays the text in a dialog box at the center of the screen.
#
#####

$CAPSHOME/bin/xmessage.static -geometry 352x84+447+334 \
    -fn "-*-courier-bold-r-*-*-18-*-*-18-*-*-18-*-*-18-*-*" -bg gray \
    -title "CAPS ALERT" -buttons OK -default OK -file -
```

## browse\_sb.script

```
#!/bin/csh

#####
#
# Placeholder for the Software Base Module which is not implemented in
# this CAPS Release.
#
#####

echo "The Software Base is not implemented in CAPS Release 2"
```

## caps

```
#!/bin/csh -f

#####
# module: caps
# Description:
#   This script is used to activate the CAPS user interface.
#   It creates the necessary directories under $HOME/.cap
#   if the .cap directory does not exist when executing caps
# Usage:
#   caps
# Changes:
#   8/13/96 VB
#       Rename caps93d to caps_d and caps93m to caps_m
#       to decouple the file names from the version of the system.
#   8/28/95 MTS
#       Replace the call "$CAPSHOME/bin/caps93d.exe&" with
#       "($CAPSHOME/bin/caps93d.exe; rm $HOME/.caps/temp/*)&"
#       in order to clean up the .caps/temp directory
#       when quitting caps.
#
#####

unset noclobber

if (! -d $HOME/.caps) then
  echo "Making .caps directory"
  mkdir $HOME/.caps
  cp -rp $CAPSHOME/demo/* $HOME/.caps
  chmod 755 $HOME/.caps
endif

if (! -d $HOME/.caps/temp) then
  echo "Making .caps/temp directory"
  mkdir $HOME/.caps/temp
  chmod 755 $HOME/.caps/temp
endif

chmod -R u+w $HOME/.caps &

set CAPS_version = "designer"
set DDBName = $USER

setenv LD_LIBRARY_PATH $CAPS_LD_PATH

# set parameters from flags

set n = $#argv
while ($n >= 1)
  switch ($1)
    case "-m":
      shift
      set CAPS_version = "manager"
```

```

        @ n -= 1
        breaksw
    case "-d":
        shift
        set DDBName = $1
        setenv CAPS93_DDB $DDBName
        shift
        @ n -= 2
        breaksw
    endsw
end

switch ($CAPS_version)
case "designer":
    # echo "executing CAPS in designer mode"

# replace old code
# $CAPSHOME/bin/caps93d.exe&
# by the following line

    ($CAPSHOME/bin/caps_d.exe; rm $HOME/.caps/temp/*)&
    breaksw
case "manager":
    # echo "executing CAPS in manager mode"

# replace old code
# $CAPSHOME/bin/caps93d.exe&
# by the following line

    ($CAPSHOME/bin/caps_m.exe; rm $HOME/.caps/temp/*)&
    breaksw
endsw

```

## **caps\_m.exe**

```
#!/bin/csh
```

```
#####  
#  
# Placeholder for the CAPS Manager Mode Interface which is not  
# implemented in this CAPS Release. A message is echoed to the terminal  
# window and CAPS is started in the default Designer Mode.  
#  
#####
```

```
echo " "  
echo "The Manager Mode is not implemented in CAPS Release 2."  
echo "Starting CAPS in Designer Mode ..."  
$CAPSHOME/bin/caps_d.exe
```

## CAPSsetup

```
#####
#
# CAPSsetup
#
# This file sets up environment variables and library paths for CAPS to
# operate properly. It also checks if external programs used by CAPS
# are present and sets additional variables for them. Such external
# programs include Openwindows, TAE Plus, and your Ada compiler.
#
# This file should be sourced from within the user's .cshrc file. It may
# need editing to tailor it to your system's environment.
#
# Example use in a user's .cshrc file:
# (Should be placed after setups for CAPS external programs)
#
# # setup for caps
# if (-d /opt/caps) then
#     setenv CAPSHOME /opt/caps
#     set path =($path $CAPSHOME/bin)
#     source $CAPSHOME/bin/CAPSsetup
# endif
#
# NOTE: $CAPSHOME needs to be defined before sourcing CAPSsetup.
# Additionally, CAPSsetup assumes that TAE Plus and the Ada Compiler have
# already been correctly setup for use in accordance with their setup
# instructions. In Particular, $TAE and $TAEADALIB need to be set.
#
#####

#####
# Tool location definitions
#####

set TYPE = `uname -r`
if ($TYPE:r:r == "5") then
# We are on a Solaris version 2.x system - set paths for Solaris

# path to Open Windows
setenv OPENWINHOME /usr/openwin

# path to the Sun/Verdix Ada root directory
setenv ADA_HOME /opt/vads

# path to Vads Ada editor
setenv AEDIT_BASE /opt/VADSedit
else
# We are on SunOS version 4.x - set paths for SunOS

# path to Open Windows
setenv OPENWINHOME /usr/openwin

# path to the Sun/Verdix Ada root directory
```



```

        setenv ADA_HOME /usr/local/SunAda

# path to Vads Ada editor
        setenv AEDIT_BASE /usr/local/VADSedit

endif

#####
# Do not modify anything below this line
#####

#
# CAPS
#
if (-d $CAPSHOME) then
    setenv CAPS_COMPILER_FLAGS -lX11
    setenv CAPS_DDB $user
    setenv CAPS_CPU_RATIO 1.0
    set path = ($CAPSHOME/bin $path)
else
    echo "Environment variable CAPSHOME is not set properly,"
    echo "See CAPS Installation Guide"
    exit
endif

#
# OPENWIN
#
if (-d $OPENWINHOME) then
    # set path = ($path $OPENWINHOME/bin)
    set MOTIF_LD_LIB = ":$OPENWINHOME/lib"
else
    echo "Could not find openwin"
    set MOTIF_LD_LIB = ""
endif

#
# Ada compiler
#

#
# Vads Self Ada Compiler
#
if (-d $ADA_HOME) then
    set ADA_LD_LIB = ""
    set path = ($ADA_HOME/bin $path)
else
    echo Ada compiler not found
    set ADA_LD_LIB = ""
endif

#
# TAE+
#

```

```

# set up for TAE Plus 5.31 for Solaris
if (-d $TAE) then
    set TAE_LD_LIB = ":$TAEADALIB"
else
    echo TAE not found or not properly setup
    set TAE_LD_LIB = ""
endif

#
# VADSedit
#
if (-d $AEDIT_BASE) then
    set path = ($AEDIT_BASE/bin $path)
endif

#
# set up dynamic initialization for LD_LIBRARY_PATH
#
setenv CAPS_LD_PATH /usr/lib$ADA_LD_LIB$TAE_LD_LIB$MOTIF_LD_LIB

#
# ALIASES
#
if (-d /opt) then
# for Solaris
    alias lpr 'lp'
endif

```

## compile.script

```
#!/bin/csh -f

#####
# module: compile.script
# Description:
#   This script is used by CAPS to compile a prototype.
#   Assumes $ADA_HOME, $TAE, and $TAEADALIB previously set by
#   CAPSsetup and the user's TAE setup.
# Usage:
#   compile.script $HOME prototype_name v.v
# Inputs:
#   Files:
#     prototypeDir/prototypeName.*a
# Outputs:
#   Files:
#     prototypeDir/ada subdirectory created if necessary
#     prototypeDir/bin subdirectory created if necessary
#     executable is prototypeDir/bin/prototypeName.exe
#
# Changes:
#   9/23/96 DMI
#     Added entire path for compiler related commands by using the
#     $ADA_HOME variable to make script independent of path setup.
#   8/31/96 DMI
#     Changed steps to determine correct compiler version to work with
#     versions 1.1 and higher of the VADSSelf Ada Compiler.
#     Set a.path to "-a $TAEADALIB -L $prototypeDir/ada" instead of
#     "-a $TAE/ada/lib/sun4 -L $prototypeDir/ada" for portability.
#   5/22/96 VB
#     Set tmpDir to "$prototypeDir/temp" instead of "$1/.caps/temp" to avoid
#     multiple users writing to the same file.
#   8/28/95 MTS
#     replace all occurrences of "/tmp/compile.script.temp"
#     by $tmpDir/compile.script.temp"
#   5/22/95 MTS
#     Set tmpDir to "$1/.caps/temp" instead of "/tmp" to avoid
#     multiple users writing to the same file.
#
#####

echo "Starting CAPS Compilation Subsystem"

#####
# initialize parameters

unset noclobber

set prototypeName = $2
set prototypeDir = $1/.caps/$2/$3

set tmpDir = "$prototypeDir/temp"

# make sure the temporary directory exists
```

```

if (! -d $tmpDir) then
    echo "creating temporary directory"
    mkdir $tmpDir
    chmod 755 $tmpDir
endif

cd $prototypeDir

if ($?CAPS_COMPILE_FLAGS) then
    set compileFlags = $CAPS_COMPILE_FLAGS
else
    set compileFlags = ""
endif

#####
# Check to see if an ada subdirectory exists in the current prototype
# directory. If not, make one.

if (! -d $prototypeDir/ada) then
    echo "Making Ada library in "$prototypeDir
    mkdir $prototypeDir/ada
    chmod 755 $prototypeDir/ada

    echo "q" | $ADA_HOME/bin/a.mklib -i $prototypeDir/ada >
$tmpDir/compile.script.temp

    #Look for correct compiler version
    set line = `grep $ADA_HOME $tmpDir/compile.script.temp`

    rm $tmpDir/compile.script.temp

    if (" $line" != "") then
        # Strip off ending period if necessary
        set choice = `expr $line[1] : '\([0-9]*\)'.`
        echo $choice | $ADA_HOME/bin/a.mklib -i $prototypeDir/ada >& /dev/null
    else
        echo "Ada compiler not installed"
    endif

    $ADA_HOME/bin/a.path -a $CAPSHOME/lib -L $prototypeDir/ada

    if ($?TAE) then
        $ADA_HOME/bin/a.path -a $TAEADALIB -L $prototypeDir/ada
    endif
endif

#####
# Check for a bin directory in which to put the executable

if (! -d $prototypeDir/bin) then
    echo "Making bin directory "$prototypeDir/bin
    mkdir $prototypeDir/bin
    chmod 755 $prototypeDir/bin
endif

```

```
#####
# Compile

echo "Compiling prototype "$prototypeName
cd ada
$ADA_HOME/bin/a.make -v -w $prototypeName $compileFlags \
    -o ../bin/$prototypeName.exe -f ../$prototypeName.*a

#####
# alert compilation complete

echo "Compilation complete" | alert
```

## edit\_ada.script

```
#!/bin/csh -f

#####
# module: edit_ada.script
# Description:
#   This script is used by CAPS to invoke the Verdix Syntax Directed
#   Ada Editor.
# Usage:
#   edit_ada.script $HOME prototype_name v.v
# Inputs:
#   Files:
#     prototypeDir/prototypeName.*a
# Outputs:
#   Files:
#     Any modified file in prototypeDir
#
#####

cd $1/.caps/$2/$3

if (${?AEDIT_BASE}) then
  setenv AEDIT_PATH  $AEDIT_BASE/configuration
  setenv AEDIT_BIN   $AEDIT_BASE/bin
  setenv AEDIT_XCHARS $AEDIT_PATH/kyn_xchars
  setenv AEDIT_XKEYS  $AEDIT_PATH/kyn_xkeys.s
  setenv AEDIT_XMENU  $AEDIT_PATH/kyn_xmenu.vdx.s
#
  $AEDIT_BIN/vads_edit $AEDIT_BIN/keyflex -showButtons $*
else
  echo "No Ada syntax directed editor installed"
  echo "Edit caps defaults and choose a text editor instead."
endif
```

## edit\_interface.script

```
#!/bin/csh -f

#####
# module: edit_interface.script
# Description:
#   This script is used by CAPS to invoke TAE+ to build a prototype
#   user interface.
# Usage:
#   edit_interface.script $HOME prototype_name v.v
# Inputs:
#   Files:
#     prototypeDir/bin/prototypeName.res (TAE+ resource file)
#     prototypeDir/bin/<TAE+ picture files>
# Outputs:
#   Files:
#     prototypeDir/bin/prototypeName.res (TAE+ resource file)
#     prototypeDir/bin/<TAE+ picture files>
#
# Changes:
#   8/11/96 VB
#     change the third parameter from v.v/bin to just v.v
#     for uniformity and because the bin is easy to put on and
#     hard to take off.
#   7/30/96 VB
#     add calls to automatic tae code transformation/genertion routines.
#   5/22/96 VB
#     Set tmpDir to "$prototypeDir/temp" instead of "$1/.caps/temp" to avoid
#     multiple users writing to the same file.
#   5/22/95 MTS
#     Set tmpDir to "$1/.caps/temp" instead of "/tmp" to avoid
#     multiple users writing to the same file.
#####

#####
# initialize parameters
#####

unset noclobber

set directory = $1
set prototypeName = $2
set version = $3

set prototypeDir = $directory/.caps/$prototypeName/$version
set tmpDir = "$prototypeDir/temp"

# make sure the temporary directory exists
if (! -d $tmpDir) then
  echo "creating temporary directory"
  mkdir $tmpDir
  chmod 755 $tmpDir
endif
```

```

#####
# Check for a bin directory in which to put the executable
#####

if (! -d $prototypeDir/bin) then
  echo "Making bin directory "$prototypeDir/bin
  mkdir $prototypeDir/bin
  chmod 755 $prototypeDir/bin
endif

#####
# Change to the prototypeDir/bin directory and invoke TAE+
#####

cd $prototypeDir/bin
taewb $prototypeName

#####
# If a prototypeName.a file has been created using TAE+,
# convert it to CAPS interface conventions,
# generate interface files, and copy them up to the prototype directory,
# leaving the TAE+ resource file
# and any picture files in the prototypeDir/bin directory.
#####

if (-f ${prototypeName}.a) then
  echo ""
  echo "Converting TAE+ generated file to CAPS format"
  tae_to_caps $prototypeName $version
  echo "TAE+ conversion complete."
  mv ${prototypeName}.a ${prototypeName}.a.generated
endif

```



## edit\_psd1.script

```
#!/bin/csh -f

#####
# module: edit_psd1.script
# Description:
#   This script is used by CAPS to invoke the PSDL Editor
# Usage:
#   edit_psd1.script $HOME prototype_name v.v
# Inputs:
#   Files:
#     prototypeDir/prototypeName.psd1
# Outputs:
#   Files:
#     prototypeDir/prototypeName.psd1
#
# Changes:
#   5/22/96 VB
#     Set tmpDir to "$prototypeDir/temp" instead of "$1/.caps/temp" to avoid
#     multiple users writing to the same file.
#   5/22/95 MTS
#     Set tmpDir to "$1/.caps/temp" instead of "/tmp" to avoid
#     multiple users writing to the same file.
#
#####

#####
# initialize parameters

unset noclobber

set prototypeName = $2
set prototypeDir = $1/.caps/$2/$3

set tmpDir = "$prototypeDir/temp"

# make sure the temporary directory exists
if (! -d $tmpDir) then
  echo "creating temporary directory"
  mkdir $tmpDir
  chmod 755 $tmpDir
endif

#####
# Change to the prototypeDir directory, prepare the attribute file name
# file and invoke the Editor

cd $prototypeDir
rm -f attr_file_name.grf
rm -f gedatatransfile.txt
rm -f gedatatransfile2.txt
echo > gedatatransfile.txt
echo > gedatatransfile2.txt
echo $prototypeName.grf > attr_file_name.grf
```

```
echo $prototypeName.grf >> attr_file_name.grf
if (! -f $prototypeName.grf) then
  echo > $prototypeName.grf
endif
psdl_editor -geom 600x750+0-0 $2.psdl
```

```
#####
# Clean up.
```

```
rm attr_file_name.grf
rm gedatatransfile.txt
rm gedatatransfile2.txt
rm free_list.stats
```

## execute.script

```
#!/bin/csh -f

#####
# Module: execute.script
# Description:
#   This script is used by CAPS to execute the prototype in a separate
#   terminal window.
# Inputs:
#   Files:
#     prototypeDir/prototypeName.exe is the executable file
#     There may be other inputs in this directory: application dependent
# Outputs:
#   Files:
#     application dependent
#
#
# Changes:
#   5/22/96 VB
#     Set tmpDir to "$prototypeDir/temp" instead of "$1/.caps/temp" to avoid
#     multiple users writing to the same file.
#   5/22/95 MTS
#     Set tmpDir to "$1/.caps/temp" instead of "/tmp" to avoid
#     multiple users writing to the same file.
#
#####

#####
# initialize parameters

unset noclobber
set prototypeDir = $1/.caps/$2/$3

set tmpDir = "$prototypeDir/temp"

# make sure the temporary directory exists
if (! -d $tmpDir) then
    echo "creating temporary directory"
    mkdir $tmpDir
    chmod 755 $tmpDir
endif

set prototypeName = $2

#####
# execute the prototype in a separate xterm

echo "CPU speed ratio = " $CAPS_CPU_RATIO

cd $prototypeDir/bin
xterm -n $prototypeName -g 80x25+10+300 -sb -e ./ $prototypeName.exe
```

## make.script

```
#!/bin/csh -f

#####
# Module name: make.script
# Description:
#   This script is used by CAPS to execute the scheduler and
#   create the prototype.a file.
# Usage: make.script <home_directory> <prototype_name> <variation.version>
# Inputs:
#   Files:
#     $1/.caps/temp/<prototype_name>.psdl (The complete PSDL program)
#     From the translator: prototypeDir/exceptions.a
#                           prototypeDir/timers.a
#                           prototypeDir/streams.a
#                           prototypeDir/main.a
#                           prototypeDir/drivers.a
#     scheduler uses stdin to select scheduling algorithm. This
#     could come from the user, but in this implementation, the
#     scheduling algorithms are automatically tried one at a time.
#     From scheduler: prototypeDir/ss.a
#                     prototypeDir/ds.a
# Outputs:
#   Files:
#     prototypeDir/<prototype_name>.a
#
# Changes:
#   5/22/96 VB
#     Set tmpDir to "$prototypeDir/temp" instead of "$1/.caps/temp" to avoid
#     multiple users writing to the same file.
#   5/22/95 MTS
#     replace all reference to "/tmp" by "$1/.caps/temp"
#   8/21/95 MTS
#     change the condition "$next <= 4" to "$next <= 2"
#     and the condition "$next > 4" to "$next > 2"
#     to skip the simulated annealing algorithm
#####
# initialize parameters

unset noclobber
set schedAlgorithm = 1
set prototypeDir = $1/.caps/$2/$3

set tmpDir = "$prototypeDir/temp"

# make sure the temporary directory exists
if (! -d $tmpDir) then
  echo "creating temporary directory"
  mkdir $tmpDir
  chmod 755 $tmpDir
endif

set prototypeName = $2
```

```

cd $prototypeDir

#####
# check for psdl file and output from the translator

if (! -f $tmpDir/$prototypeName.psdl) then
    echo "    - PSDL source file" $tmpDir/$prototypeName.psdl "not found.
Aborting."
    exit(-1)
endif

if (! -f main.a) then
    echo "Prototype Translation Required" | alert
    exit(-1)
endif

#####
# Execute the scheduler

echo " - Starting CAPS Scheduler ..."

# pre_ss $tmpDir/${prototypeName}.psdl | decomposer > atomic.info

set next = 1

set ssOut = `echo $next "\nq\n" | scheduler $tmpDir/$prototypeName.psdl
$prototypeName.diag | grep "schedule found"`

# replace old code
#     while ($next <= 3 && $#ssOut == 0)
# by the following line to skip the simulated annealing algorithm

while ($next <= 2 && $#ssOut == 0)
    if ($next != $schedAlgorithm) then
        echo "    - Feasible schedule not found. Trying algorithm" $next
        set ssOut = `echo $next "\nq\n" | scheduler $tmpDir/$prototypeName.psdl
$prototypeName.diag | grep "schedule found"`
    endif
    @ next ++
end

# replace old code
#     if ($next > 3) then
# by the following line to match the above correction

if ($next > 2) then
    echo "    - Feasible schedule not found. Aborting."
    exit(-1)
    echo "    - Deleting Temporary Files"
    rm -f atomic.info
    rm -f non_crits
    rm -f tl.a
    rm -f ds.a
    rm -f ss.a
    rm -f stdin.psdl.lst

```

```

rm -f free_list.stats
rm -f exceptions.a
rm -f instantiations.a
rm -f timers.a
rm -f streams.a
rm -f drivers.a
rm -f main.a
endif
echo "    -" $ssOut

#####
# create prototypeName.a file

echo " - Generating "$prototypeName.a
if (-f $prototypeName.a) then
    cp $prototypeName.a $tmpDir
endif

#####
# Put the exceptions, timers and streams packages in

echo "    - Installing Exceptions Package"
cat exceptions.a > $prototypeDir/$prototypeName.a

echo "    - Installing Instantiations Package"
cat instantiations.a >> $prototypeDir/$prototypeName.a

echo "    - Installing Timers Package"
cat timers.a >> $prototypeDir/$prototypeName.a

echo "    - Installing Streams Package"
cat streams.a >> $prototypeDir/$prototypeName.a

#####
# Add the drivers package to <prototype_name>.a

echo "    - Installing Drivers Package"

cat drivers.a >> $prototypeDir/$prototypeName.a

#####
# cat the dynamic shedule to <prototype_name>.a

echo "    - Installing Dynamic Schedule Package"
cat ds.a >> $prototypeDir/$prototypeName.a

#####
# cat the static shedule to <prototype_name>.a

echo "    - Installing Static Schedule Package"
cat ss.a >> $prototypeDir/$prototypeName.a

#####

```

```

# Add the final with/use lines and begin null; end; to <prototype_name>.a

echo "    - Installing main procedure"
cat main.a >> $prototypeDir/$prototypeName.a

#####
# remove temp files

echo "    - Deleting Temporary Files"
rm -f atomic.info
rm -f non_crits
rm -f tl.a
rm -f ds.a
rm -f ss.a
rm -f stdin.psd1.lst
rm -f free_list.stats
rm -f exceptions.a
rm -f instantiations.a
rm -f timers.a
rm -f streams.a
rm -f drivers.a
rm -f main.a

#####
# alert build complete

echo "Scheduling complete" | alert

```

## save\_ada\_files

```
#!/bin/csh

#####
#
#   This script saves copies of the old versions of all ada files
#
#####

set dir = $1

cd $dir

set nonomatch

set pattern = "*.a"
set files = $pattern
if ( "$files" != "$pattern" ) then
    # avoid a failure if the pattern driving the loop does not match anything.
    foreach file ($files)
        cp -p $file $file.bak
    end
endif

unset nonomatch
```



## show\_ops.exe

```
#!/bin/csh

#####
#
# Placeholder for the Design Database Module which is not implemented in #
# this CAPS Release.
#
#####

echo "The Design Database is not implemented in CAPS Release 2"
```

## tae\_to\_caps

```
#!/bin/csh

#####
#
# This script converts TAE generated Ada code to run in
# the CAPS environment. It assumes the Ada code was generated
# using the multi-file style, with the option
# "Generate default print statements in Event Handlers" DISABLED.
#
# This program reads the "<prototype>.a" file generated by TAE and
# generates the files needed to interface to CAPS. The main file
# "<prototype>.generated_tae_event_monitor.a" contains an Ada package
# with a procedure that performs one cycle of the TAE event loop with
# has bounded execution time. The command line to invoke the script is:
#
#   tae_to_caps <prototype> <version>
#
# This script should be called only if the file <prototype>.a exists
#
#####

# check that the shell script has the right number of parameters
if ( $#argv != 2 ) then
    echo "Usage: tae_to_caps <prototype> <version>"
    exit
endif

# unpack the parameters
set prototype=$1          # the name of the prototype
set version=$2            # variation.version

# define local variables
set capsPath=$HOME/.caps/$prototype/$version
# the path for the prototype directory
set caps_lib_path=$CAPSHOME/bin
# the path for finding the awk scripts
set caps_temp_path=$capsPath/temp
# the path for the temporary files
set inDeclFile=$caps_temp_path/input_items
# the description file for tae input items
set outDeclFile=$caps_temp_path/output_items
# the description file for tae output items

set adaFile=$capsPath/bin/$prototype.a
# the ada code file generated by TAE+

set monitor_file=$capsPath/$prototype.generated_tae_event_monitor.a
# the output file for the generated event monitor procedure
set task_file=$capsPath/$prototype.event_task.a
# the output file for the generated task

# make sure the temporary file directory exists
if ( ! -d $caps_temp_path ) then
```

```

    echo making temp directory
    mkdir $caps_temp_path
endif

# save backup copies of all previously existing ada files
echo "Making backup copies of existing ada files"
save_ada_files $capsPath

# Generate the description files for tae input items and tae output items
set specs = ($capsPath/*.spec.psd1)
set imps = ($capsPath/*.imp.psd1)
echo "dummy input" | nawk -f $caps_lib_path/find_items.awk \
    specs="$specs" imps="$imps" inDeclFile=$inDeclFile outDeclFile=$outDeclFile

# generate the input bubbles
if (-f $inDeclFile) then
    nawk -f $caps_lib_path/input_bubbles.awk \
        prototype=$prototype path=$capsPath \
        $inDeclFile
else
    echo "The description file for tae input items does not exist"
endif

# generate the input monitor procedure
echo generating $monitor_file
nawk -f $caps_lib_path/event.awk \
    $adaFile > $monitor_file

if (-f $outDeclFile) then
    # generate the output bubbles
    nawk -f $caps_lib_path/output_bubbles.awk \
        prototype=$prototype path=$capsPath $outDeclFile

    # generate the task that provides mutual exclusion for calls to TAE
    echo generating $task_file
    nawk -f $caps_lib_path/task.awk \
        prototype=$prototype $outDeclFile > $task_file

    # generate the output procedure specs in the panel package spec
    nawk -f $caps_lib_path/panel_specs.awk \
        prototype=$prototype capsPath=$capsPath inDeclFile=$inDeclFile \
        $outDeclFile
else
    echo "The description file for tae input items does not exist"
endif

# generate the augmented panel bodies
if (-f $inDeclFile && -f $outDeclFile) then
    nawk -f $caps_lib_path/panel_bodies.awk \
        prototype=$prototype capsPath=$capsPath outDeclFile=$outDeclFile
    $inDeclFile
endif

# copy up all generated and unmodified ada files
cd $capsPath/bin

```

```
echo generating $capsPath/${prototype}.global_b.a
cp global_b.a ../${prototype}.global_b.a

echo generating $capsPath/${prototype}.global_s.a
cp global_s.a ../${prototype}.global_s.a

echo generating $capsPath/${prototype}.${prototype}__creat_init.a
cp ${prototype}__creat_init.a ../${prototype}.${prototype}__creat_init.a

echo generating $capsPath/${prototype}.${prototype}_support_b.a
cp ${prototype}_support_b.a ../${prototype}.${prototype}_support_b.a

echo generating $capsPath/${prototype}.${prototype}_support_s.a
cp ${prototype}_support_s.a ../${prototype}.${prototype}_support_s.a
```

## temp\_dir.script

```
#!/bin/csh -f

#####
# module: temp_dir.script
# Description:
#   This script is used by CAPS to check if the prototype's
#   temporary directory exists, and if not, creates it.
# Usage:
#   temp_dir.script $prototypeDir/temp
# Inputs:
#   Files:
#     none
# Outputs:
#   Files:
#     Creates prototypeDir/temp if needed
#
#####

set tmpDir = $1

# make sure the temporary directory exists
if (! -d $tmpDir) then
  echo "creating temporary directory"
  mkdir $tmpDir
  chmod 755 $tmpDir
endif
```

## translate.script

```
#!/bin/csh -f

#####
# module: translate.script
# Description:
#   This script is used by CAPS to run the translator.
# Usage: translate.script $HOME <prototype_name> v.v
# Inputs:
#   Files:
#       expander uses prototypeDir/<prototype_name>.psdl
#       translator uses tmpDir/<prototype_name>.psdl
# Outputs:
#   Files:
#       expander writes to tmpDir/prototype.psd1
#       translator writes to:
#       prototypeDir/streams.a
#       prototypeDir/timers.a
#       prototypeDir/exceptions.a
#       prototypeDir/main.a
#       prototypeDir/drivers.a
# Changes:
#   5/22/96 VB
#       Set tmpDir to "$prototypeDir/temp" instead of "$1/.caps/temp" to avoid
#       multiple users writing to the same file.
#   5/22/95 MTS
#       Set tmpDir to "$1/.caps/temp" instead of "/tmp" to avoid
#       multiple users writing to the same file.
#
#####
# initialize parameters

unset noclobber
set debug = ""
set prototypeName = $2
set prototypeDir = $1/.caps/$2/$3

set tmpDir = "$prototypeDir/temp"

# make sure the temporary directory exists
if (! -d $tmpDir) then
    echo "creating temporary directory"
    mkdir $tmpDir
    chmod 755 $tmpDir
endif

cd $prototypeDir

#####
# remove temporary files

rm -f $tmpDir/$prototypeName.err

#####
```

```

echo " - Expanding PSDL source ..."
echo "   - Expanding" $prototypeName.psdl

(cat $prototypeName.psdl | expander >! $tmpDir/$prototypeName.psdl) >&!
$tmpDir/$prototypeName.err

#####
# if expander got errors, alert the user

set err = `cat $tmpDir/$prototypeName.err`
if ($#err > 1) then
    cat $tmpDir/$prototypeName.err
    echo Syntax errors found during expansion. See $prototypeName.psdl | alert
    exit(-1)
endif

#####
# if there were no expander errors, execute translator

echo " - Translating PSDL source ..."
echo "   - Generating CAPS prototype packages"

$CAPSHOME/bin/translator.exe $tmpDir/${prototypeName}.psdl

#####
# remove temporary files

rm -f free_list.stats

#####
# alert translation complete

echo "Translation complete" | alert

```

## **APPENDIX C. WORLD WIDE WEB SITES**

The following is a list of World Wide Web Uniform Resource Locators (URLs) frequently visited during this thesis research. They provide the latest information on the software components used in the CAPS operating environment. They are an ideal starting point for product information, questions, troubleshooting, and future research. These web locations were valid at the time of this writing; however, due to the ever changing nature of the web, some may no longer be valid.

### **CAPS Research Project**

<http://wwwcaps.cs.nps.navy.mil/>

### **Solaris**

#### **Sun Microsystems**

<http://www.Sun.COM:80/index.html>

#### **Sun Microsystems Computer Company**

<http://www.sun.com/corporateoverview/smcc/index.html>

#### **Sun Products and Solutions**

<http://www.sun.com/products-n-solutions/index.html>

#### **Solaris Products**

<http://www.sun.com/solaris/index.html>

#### **Solaris Products Site Map - Good starting point for Solaris information**

<http://www.sun.com/solaris/sitemap.html>

#### **Solaris 2.5 Overview**

<http://www.sun.com/solaris/2.5/index.html>

#### **Solaris 2.5.1 Overview**

<http://www.sun.com/solaris/2.5/2.5.1/index.html>

#### **Solaris Migration Initiative Home Page**

<http://www.sun.com/smcc/solaris-migration/>



Solaris Migration Initiative Home Page

<http://www.sun.com:80/smcc/solaris-migration/docs/migration-initiative.html>

Help Migrating from SunOS to Solaris 2.X

<http://www.swcp.com/~pcaskey/sunos-migrate.html>

Solaris 1.x to Solaris 2.x Transition Guide

<http://www.sun.com/smcc/solaris-migration/docs/guide-files/transition-guide.html>

Solaris 2 Frequently Asked Questions

<http://gonzo.tamu.edu/solaris2.html>

Solaris 2 Porting FAQ

<http://www.cis.ohio-state.edu/hypertext/faq/usenet/Solaris2/porting-FAQ/faq.html>

SunSoft Support Resolutions

<http://access1.sun.com/>

SunSoft Solaris Support

<http://access1.sun.com/TechRoom/solaris.html#sparc>

SunSolve ONLINE

<http://192.9.9.24/>

FAQ for the OPEN LOOK User Interface

[http://www.cs.indiana.edu/faq/OpenLook/front\\_page.html](http://www.cs.indiana.edu/faq/OpenLook/front_page.html)

Sun Books

<http://www.sun.com/cgi-bin/show?smcc/solaris-migration/docs/books.html>

## **TAE Plus**

TAE Plus

[http://groucho.gsfc.nasa.gov/Code\\_520/Code\\_522/Projects/TAE/](http://groucho.gsfc.nasa.gov/Code_520/Code_522/Projects/TAE/)

TAE Plus Home Page

<http://www.cen.com/tae/taehome.html>

Century Computing, Inc. Home Page

<http://www.cen.com/>

## **OSF / Motif**

OSF Home Page

<http://www.osf.org/>

User Environment Information

<http://www.osf.org/motif/index.html>

OSF/Motif® 2.0 Information

<http://www.osf.org/motif/Motif20/index.html>

Motif FAQ

<http://www.cis.ohio-state.edu/hypertext/faq/usenet/motif-faq/top.html>

Motif on the World Wide Web

<http://www.cen.com/mw3/>

Kenton Lee: Technical X Window System and OSF/Motif WWW sites

<http://www.rahul.net/kenton/xsites.html>

Integrated Computer Solutions (third-party provider of Motif binaries)

<http://www.ics.com/>

## **X Windows**

X Consortium Home Page

<http://www.x.org/>

X Window System, Release 6

<http://www.x.org/consortium/R6index.html>

X11 Release 6, Release Notes

<http://www.x.org/consortium/R6doc/relnotes/>

X11 FAQ

<http://www.cis.ohio-state.edu/hypertext/faq/usenet/x-faq/top.html>

Kenton Lee: Technical X Window System and OSF/Motif WWW sites

<http://www.rahul.net/kenton/xsites.html>

Getting X11R6

<http://www.x.org/consortium/GettingX11R6.html>

Walnut Creek CDROM (more than just X11)

<http://www.cdrom.com>

X11R6 Source Code From Walnut Creek CDROM

<http://www.cdrom.com/pub/X11R6/>

X11 Contrib FAQ

<ftp://ftp.x.org/contrib/faqs/FAQ>

### **Synthesizer Generator**

GrammaTech, Inc.

<http://www.grammatech.com/>

### **Eli**

University of Colorado at Boulder

<http://www.cs.colorado.edu/~eliuser/>

### **Alternate User Interface Tools**

User Interface Software Tools

<http://www.cs.cmu.edu/afs/cs/user/barn/www/toolnames.html>

Fresco

<http://www.faslab.com/fresco/HomePage.html>

TCL WWW Info

<http://www.sco.com/Technology/tcl/Tcl.html>

Togl

<http://www.ssec.wisc.edu/~brianp/Togl.html>

## **Ada Programming Language and Compilers**

Ada Information Clearinghouse Home Page

<http://sw-eng.falls-church.va.us/AdaIC/Welcome.html>

Ada 95 Information

[http://www.acm.org/sigada/ada\\_95/ada\\_95.html](http://www.acm.org/sigada/ada_95/ada_95.html)

Home of the Brave Ada Programmers

<http://lglwww.epfl.ch/Ada/>

Public Ada Library (PAL)

<http://www.cdrom.com/pub/ada/>

Lovelace Ada Tutor Home Page

<http://lglwww.epfl.ch/Ada/Tutorials/Lovelace/lovelace.html>

Ada 95 Reference Manual

<http://lglwww.epfl.ch/Ada/LRM/9X/rm9x/rm9x-toc.html>

The Free X11/Ada95 Programmer's home Page

<http://ocsystems.com/xada/>

AdaCore Home Page (Gnat Ada Compiler)

<http://www.gnat.com/>

Download Gnat Ada Compiler

<ftp://cs.nyu.edu/pub/gnat/>

Rational Software Corporation (VADS, Apex, Spire Compilers)

<http://www.rational.com/>

Intermetrics (AdaMagic and AppletMagic)

<http://www.inmet.com/MID/index.html>

AdaMagic (Ada95 Compiler)

<http://www.inmet.com/MID/WHAT/adamagic.html>

AppletMagic® (Ada95 to Java Compiler)

<http://www.inmet.com/MID/WHAT/java.html>

Download AppletMagic®

<http://www.inmet.com/javadir/download/>

Thomson Software - Object Ada

<http://www.thomsoft.com/products/ada/Ada.html>

GRASP Home Page (Gnat Compiler front end and more)

<http://www.eng.auburn.edu/departement/cse/research/grasp/grasp.html>

## **Perl**

The Perl Language Home Page

<http://www.perl.com/perl/index.html>

Perl Manual

<http://perl.com/perl/manual/>

Index of Perl/HTML archives

<http://www.seas.upenn.edu/~mengwong/perlhtml.html>

## **Miscellaneous Unix Information**

The Unix Reference Desk

<http://www.eecs.nwu.edu/unix.html#x11>

Unix Guru Universe

<http://www.ugu.com/sui/ugu/show?ugu>

Solaris 2.5 Freeware

[http://smc.vnet.net/solaris\\_2.5.html](http://smc.vnet.net/solaris_2.5.html)

The Internet goodies

<http://www.ensta.fr/internet/>

## **Reference Books**

Computer Literacy Bookshops  
<http://www.clbooks.com/>

O'Reilly Home Page  
<http://www.ora.com/>



## **APPENDIX D. POINTS OF CONTACT**

The following is a list of persons that were contacted during the conduct of this thesis for product information, ordering, and support:

### **Solaris:**

Betty Bennett  
Sales Representative  
Sun Microsystems Computer Company  
1842 N. Shoreline Blvd., MS UMTV80-01  
Mountain View, CA 94043-1100  
Phone: (415) 960-4404  
Fax: (415) 961-4872  
email: betty.bennett@west.sun.com

Gregory Hansen  
Systems Engineer  
Sun Microsystems Computer Company  
1842 N. Shoreline Blvd., MS MTV80-01  
Mountain View, CA 94043-1100  
Phone: (415) 960-4213  
Fax: (415) 961-4872  
email: greg.hansen@west.sun.com

### **TAE Plus:**

Lisa C. Koons  
Century Computing  
8101 Sandy Spring Road  
Laurel, MD 20707  
Phone: (301) 953-3330  
Fax: (301) 953-2368  
email: lkoons@cen.com



**Motif:**

Kathie Dawe  
Supervisor, Direct Channels  
Open Software Foundation  
11 Cambridge Center  
Cambridge, MA 02142  
Phone: (617) 621-8866  
Fax: (617) 621-0306  
email: kdawe@osf.org

**VADSSelf / Rational Apex / Spire:**

Ted Driscoll  
Rational Software Corporation  
2800 San Tomas Expressway  
Santa Clara, CA 95051-0951  
Phone: (415) 824-1430  
Fax: (415) 824-1435

**Object Ada:**

Alfred E. Miller  
Account Manager  
Thomson Software Products  
10251 Vista Sorrento Parkway  
San Diego, CA 92121  
Phone: (619) 457-2700  
Fax: (619) 452-1334  
email: amiller@thomsoft.com

**Synthesizer Generator:**

Susan Matteson  
Manager of Business Services  
GrammaTech, Inc.  
One Hopkins Place  
Ithaca, NY 14850  
Phone: (607) 273-7340  
Fax: (607) 273-8752



## INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2. Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3. Dr. Ted Lewis, Chairman, Code CS/Lt Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
4. Prof. Luqi, Code CS/Lq Computer Science Department Naval Postgraduate School Monterey, CA 93943	20
5. Prof. Berzins, Code CS/Be Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
6. Lt Dennis M. Irwin 2134 Hoyt DR Baton Rouge, LA 70816	2